パソコンプログラマのためのクリエイティブマガジン

月刊プログラマーズ・ページ

平成3年4月2日第三種郵便物認可　1992年1月1日発行(毎月1回18日) 第2巻第1号通巻13号.

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

1992 1 JANUARY

定価980円

特集

プログラミングツール大集合

そのプロフィールと使いこなしのコツ

特別企画 アンケート 著者の使っている開発ツール

創刊1周年記念

読者プレゼント

98NOTE・PS/55note

Macintosh ClassicII

が当たる!

付録:5"2HDフロッピーディスク

本誌掲載プログラム／綴じ込み付録テキストファイル(ANSI C関数一覧)／クイズ・ランダムスクリーン解読ツール／テキストファイル高速ページャー「MIEL」／Cコンパイラ「LSI C-86試食版」

SE

Shoeisha Co., Ltd.

創造をクレオらしさと呼びたい

CREO

Hello, True BASIC.

現代の正統派言語　選ぶなら世界標準

1964年、J.G.ケメニーとT.E.カーツの開発した教育用会話型言語「BASIC」は、コンピュータを大変身近なものにしました。すっきりした記述で初めての人でもわかりやすく、慣れればさらに高度に使いこなせる。当時としては、画期的な言語の登場です。こうして生まれ、多くの支持を得たBASICは、ケメニー、カーツ両氏の手によりさらなる進化を遂げました。それが、True BASICです。

①すっきりとわかりやすい、構造化プログラミング

True BASICは、コンピュータが初めての人にもわかりやすい構造化プログラミング。行番号はいっさい使わず、あなたの頭の中のアルゴリズムどおりに、プログラムは組まれます。基礎的な設計力が自然と身につくプログラミング法です。

②初めてでもとまどわない、一貫した操作性

例えば、フルスクリーンエディタとコンパイラとの完全統合。ソースコードの書きこみからデバッグ、コンパイルまで、作業は中断なく滑らかに進行。しかも、作成されたプログラムは直接実行ファイルへ変換でき、手間がかかりません。そのうえ、作業の見とおしがよいマルチウィンドウをサポート。テキストやグラフィックスの作成がスムーズに行なえます。

③シンプルで使いやすい、モジュール(機能)群

機能の高さだけでなく、操作性を重視したモジュール群。作業の流れを中断しない、系統立てた構成からなります。複雑な演算に強い、行列関数など豊富な関数。高速処理に威力を発揮する、数値演算コプロセッサ。よく使う実戦的機能の他にも、仮引数、再帰呼出処理といった高度な機能を満載しました。さらにグラフィックス機能が充実。各種ツールキットや組み込み関数を用いて、簡単作成できます。

④機種もOSも選ばない、優れた互換性

True BASICは、ANSI規格のFULL BASICに準拠。いわば世界標準のBASICです。そのうえ、ハードウェアに依存しない言語構造ですから、異なる機種間、MS-DOSからUNIXまでOS間の互換性があります。つまり、True BASICで構築されたプログラムは、特に変更せずに様々なマシン、OS上で活用できるということ。作成したプログラム資産を無駄にしません。

⑤True BASICは、すでに世界で活用中

対応機種	パソコン版	IBM	PC、PS/2
		Apple	Macintosh 512K、Plus SE、Macintosh II
		Atari	ST 520、1040、MegaST
		Commodore	Amiga 500、1000、2000
	UNIX版	Silicon Graphics Sun Microsystems, Inc. Data General Corp.	
採用している大学	全米で1600校以上		

多彩な機能がいろいろ選べる、ツールキット新発売!!

●文字列操作ツールキット
標準価格:15,000円(税別)
文字列加工からパターンマッチングまで、文字列操作に関する様々なプログラムが詰まったツールキットです。

●並べ替え・検索用ツールキット
標準価格:15,000円(税別)
文字列/数字を分類・検索するルーチンを搭載。比較ルーチン設定は4種類、複数項目の組合せ検索も可能。

●統計用グラフィックスツールキット
標準価格:25,000円(税別)
ヒストグラムや散布図、各種検定、多重線形回帰など、統計学に必要な機能を集めた本格統計ツール。

好評発売中のツールキットラインナップ

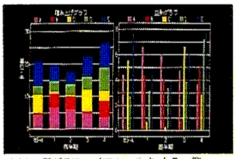
●ビジネス用グラフィックスツールキット
標準価格:18,000円(税別)

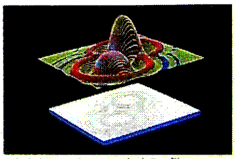
●数学者用ツールキット
標準価格:25,000円(税別)

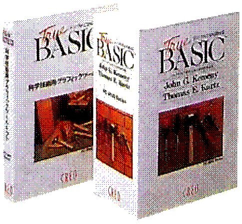
●3次元グラフィックスツールキット
標準価格:18,000円(税別)

●科学技術用グラフィックスツールキット
標準価格:25,000円(税別)

True BASIC Tool Kit


ビジネス用グラフィックスツールキットの一例


3次元グラフィックスツールキットの一例



好評発売中

True BASIC™ シンプルだから使える

PC-9800 Series
標準価格 18,000円(税別)

株式会社クレオ
〒108 東京都港区芝浦4-9-25 芝浦スクエアビル TEL03-3456-4650 FAX03-3456-4630

〈資料請求番号P02〉

"超"実力派宣言。

宣言その1

パワーで超えた。

■BASIC/98は、N88-日本語BASIC(86)(MS-DOS版)との上位互換を実現。さらに、構造化、モジュラープログラミングをサポートしています。

■EMS対応で大きなプログラムを作れ、64KB以上の配列が使えます(Pro.)。

■プルダウンメニュー方式の専用エディタがついているので、プログラムの編集・実行がスムーズに行えます(Pro.)。

宣言その2

対応機種の多さで超えた。

■BASIC/98 Pro.は5メーカー、Fastは13メーカーのパソコンに対応しています。ソフト資産が共有化できるため、作成したプログラムを無駄にすることがありません。また、対応機種間でソースプログラム互換があるので、移植も簡単です。

宣言その3

実績で超えた。

■BASIC/98には確たる実績があります。1987年発売以来、累計45,000本の販売実績と教育機関170校での採用実績です。この実績が他のBASICを圧倒しています。

■BASIC/98 Pro. インタプリタ…………27,000円
■BASIC/98 Pro. コンパイラ……………57,000円
対応機種:NEC PC−9801・EPSON PC−286/386
Panacom M・FMR/FMTOWNS・J−3100
■BASIC/98 Fast インタプリタ…………17,000円
■BASIC/98 Fast コンパイラ……………47,000円
対応機種:NEC PC−9801・EPSON PC−286/386
Panacom M・FMR/FMTOWNS・J−3100・PS/55・2020/L−720・MBC−17J・MAXY・if386AX・AX386・if 800・BS21

日本で生まれた標準BASIC

BASIC/98 Pro.

BASIC/98 Pro. にはBASIC/98 Fastのシステムディスクもはいっています。
今BASIC/98 Pro. PC-9801・EPSON PC版を買うと、近日発売予定のウィンドゥ入力ツール(サブセット版)がもれなく付いてきます。

〈資料請求番号

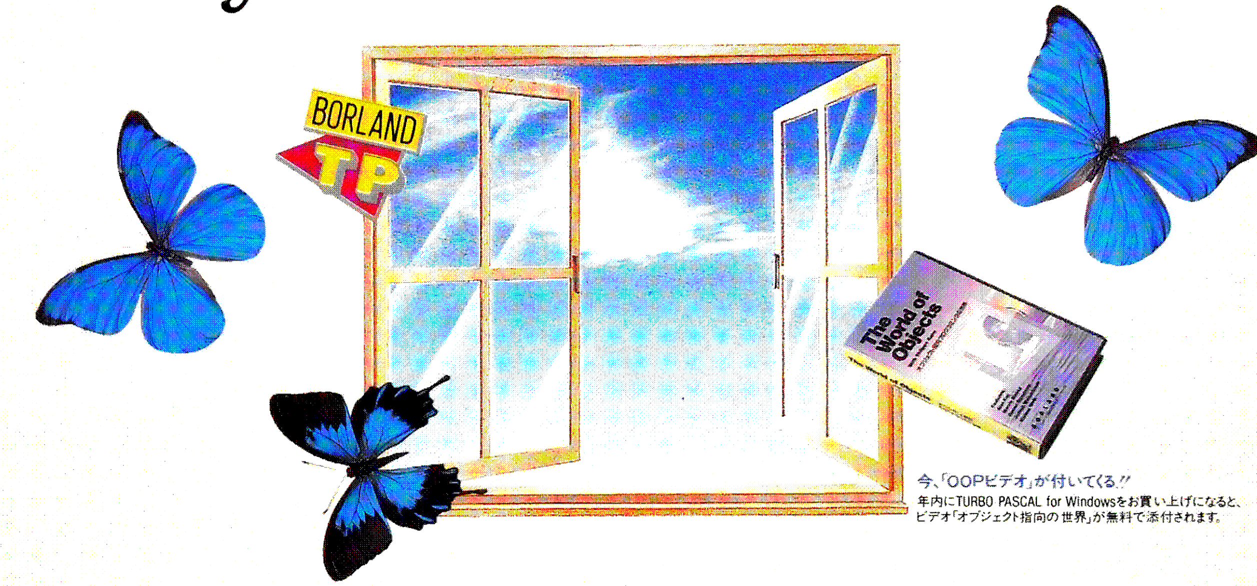
総販売元:株式会社KSP 〒107 東京都港区南青山2-9-28 青山NSビル TEL.(03)3796−8137代 FAX.(03)3796−8835

日本SEグループ

B O R L A N D

The Leader in Object-Oriented Programming

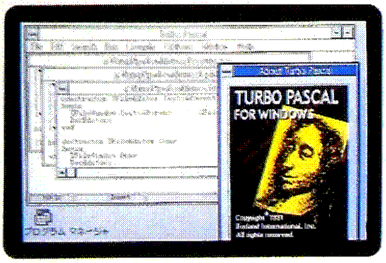
"Easy & True" for Windows.



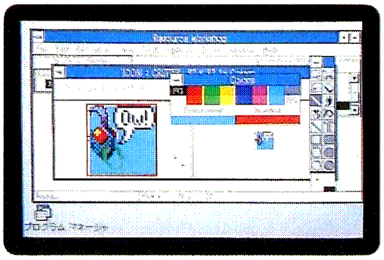
今、「OOPビデオ」が付いてくる!!
年内にTURBO PASCAL for Windowsをお買い上げになると、ビデオ「オブジェクト指向の世界」が無料で添付されます。

基本は、ターボパスカル——。
完ぺきなWindowsアプリケーション開発環境を提供。

その美しい言語仕様から"コンパイラの貴婦人"と呼ばれ、多くのプログラマーを魅了してきたパスカルが今、「Turbo Pascal for Windows」として生まれ変わりました。デビュー当時、その驚異的な低価格とコンパイルスピードの速さ、デバッガやエディタを備えた世界初の統合環境で人々を圧倒し、瞬く間に標準の地位を確立して以来、ターボパスカルは常に先進の機能を搭載し、言語の世界をリードし続けています。アメリカでは、大学などの教育機関や研究施設でターボパスカルを使用することが常識ですし、ヨーロッパや日本でもビジネスの分野で幅広く利用されています。プログラミングの学習もターボパスカルから始めれば、ベーシックで貧弱な言語で学習したために時間を無駄にする、ということはありません。しかもそのターボパスカルが、複雑で難解だったWindowsのプログラミングを簡単にしてくれるのです。



Turbo Pascal for Windowsには、Object Windowsというアプリケーションの骨格になるクラスライブラリが付属しています。ここからウィンドウやメニュー、ダイアログ、コントロールなどを継承すれば、最初からコードを記述する必要はないのです。オブジェクト指向の拡張がされていないC言語でのプログラムより、格段に素早く、短いコードでアプリケーションを作成できるようになります。Turbo Pascal for Windowsは、自分自身がWindowsのアプリケーションであり、Windowsプログラムのみの作成用に設計されています。



ソースコードの編集、コンパイル、実行、デバッグなど、使いやすさで定評のある統合環境を用い、全てWindows上で行えます。ダイアログ、メニュー、ビットマップ、カーソル、アイコン、文字列リソース、アクセレータ、フォントなどのWindowsリソースを視覚的に作成したり、編集することができるリソースワークショップも付属していますから、高額なSDKを別途購入する必要もありません。Turbo Pascal for Windowsには、Windowsアプリケーション作成に必要なツールの全てが、ひとつのパッケージに含まれているのです。

■動作環境：PC-9800シリーズ（ハイレゾモード対応、LTは除く）、PC-286/386シリーズ/MS-DOS：Ver 3.1以上/MS-Windows 3.0以上必須/マウス：バスマウス/要ハードディスク
■価格……………………29,800円（税別）

NEW

TURBO PASCAL®
FOR WINDOWS

ユーザ向け情報誌
LANGUAGE EXPRESSプレゼント!!

■この広告をご覧になって、製品に関する資料を請求していただいた方全員に、ボーランド発行の情報誌「LANGUAGE EXPRESS」をさし上げます。

＊All Borland Products are trademarks or registered trademarks of Borland International, Inc. ©Borland International, Inc.
＊Microsoft、Windowsは米国マイクロソフト社の登録商標です。その他、商品名は一般に各社の商標です。
●製品に関するお問い合せは、総販売元㈱マイクロソフトウェアアソシエイツ TEL.03-3486-1411までお願いします。

ボーランド株式会社
〒107 東京都港区南青山7-8-1 小田急南青山ビル TEL.03-3486-1400㈹
※社名変更いたしました。

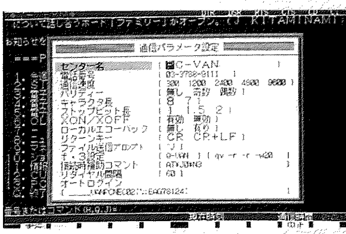
〈資料請求番号002〉

まさにプロフェッショナル!!

Yoshimasa Electronic presents MS Librarie
Microsoft. Librar

RS-232C Library　通信ソフトY-TERM装備　価格 23,000円 発売中

■製品概要　本製品は、C言語にてRS－232Cの各種制御を行うためのライブラリです。一般的な調歩同期式だけではなく、同期式についても完全にサポートされていますので、BSC等同期式手順を使うアプリケーションにも対応できます。また付属している通信ソフトY-TERMは、プログラム例としてだけでなく実際にパソコン通信等にもお使い頂くことができる機能を備えております。



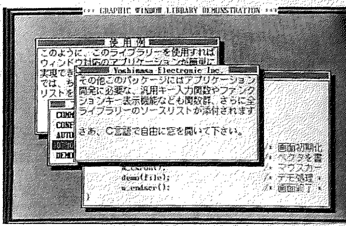
■Y-TERMの主な機能
●モデムの監視機能 ●オートログイン機能 ●自動接続機能 ●テキストファイル送受信機能 ●XMODEMファイル転送機能 ●ユーザーファンクション

■主な特長
- ●調歩同期式・同期式手順ともにサポート
- ●半／全二重制御、ハード／ソフトフロー制御のサポート
- ●拡張RS－232Cボード(PC－9861K)対応
- ●実用通信ソフトY-TERM装備
- ●Y-TERM・ライブラリともにソースコード完全公開
- ●組込みロイヤリティ無料

対応機種　NEC　PC－9801シリーズ
対応言語　MS－C　5.1以上
Quick－C　2.0以上

Graphic Window Library　グラフィック ウィンドウ ライブラリ　価格 23,000円 発売中

■製品概要　本製品は、C言語にてウィンドウ処理を行うためのライブラリです。このライブラリを使用することにより、ウィンドウ対応型アプリケーションを簡単に実現致します。ウィンドウ間の切り替えや移動等、マウスに完全対応していますのでマン・マシン・インターフェースは、抜群の威力を発揮します。またグラフィックを使用していますので、一行のドット数が可変となり、ANKフォントも自由に設定可能です。

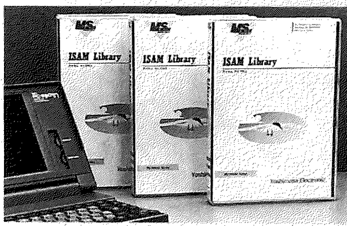


■主な特長
- ●高速グラフィックウィンドウ
- ●マウス完全対応
- ●汎用キー入力ルーチン装備
- ●ソースコード完全公開
- ●組込みロイヤリティ無料

対応機種　NEC　PC－9801シリーズ
(ハイレゾリューションモードは除く)
対応言語　MS－C　5.1以上
Quick－C　2.0以上

ISAM Library　アイサム ライブラリ　価格 23,000円 発売中

■製品概要　本製品は、C言語にてISAMファイル処理を実現するライブラリです。検索は、B－treeアルゴリズムを採用しているため、高速検索が可能です。当ライブラリにて作成のデータベースファイルは、dbaseⅢplusにて参照、変更可能です。



■主な特長
- ●B－treeによる高速アクセス
- ●ソースコード完全公開
- ●組込みロイヤリティ無料

対応機種　MS-DOSが動作可能な機種
対応言語　MS－C　5.1以上
Quick－C　2.0以上

JCA 手順 Library　通信ソフト JTRN装備　価格 46,000円 発売中

(RS-232C Libraryが含まれます)

NEWS

●OS／2対応　オンラインユーザーサポートシステム　'92年1月発売予定
全ソース付　198,000円
パソコン通信を利用した製品情報や質問回答、さらにバージョンアップ等幅広い利用が考えられます。

●弊社製品情報ネットワークシステム"Y-NET"開局中!! アクセス電話番号(03)5350-1785 USER_ID"GUEST"でどうぞ

お求め方法　弊社製品は、パソコンソフト取扱店でお求めいただけますが、直接ご注文の際は右記までお問い合わせください。
なお、上記価格には消費税が含まれておりません。

〈資料請求番号003〉

吉正電子株式会社　PC事業
〒151 東京都渋谷区代々木1-58-10 第一西脇ビ
電話 (03)5371-3041(直

Brief

Full Screen Editor 日本語版

全米のプログラマが認めた実力！
アメリカでは「ブリーフ」が
プログラミング用エディターとして
最も多く利用されています。

C言語で組めるマクロ！

Version UP!

V.3.0いよいよ登場。
プログラミングエディター新世紀。

よりパワフルで、よりエレガントな
プログラミングのために……。

●C言語で組めるマクロ 新機能

使い慣れているC言語でマクロが組めますから、習得が簡単でカスタマイズも思いのままです。

●マクロのスクリーン・デバッガ 新機能

ソースコードでデバックできます。もちろんブレークポイントやウォッチ・ドッグもセットできます。

●テンプレート機能〈構文の自動生成〉 新機能

C、Ada、Basic、COBOL、FORTRAN、MODULA2、Pascalで構文を自動生成します。

●300回ものアンドゥ

各ファイルごとに300までの操作を元にもどすことができますので、やり直しの面倒を気にせず、安心して編集できます。

●安全編集

Briefは、一定時間キーを操作しないこと最新の編集内容をファイルに自動保存します。だから編集中に電源が切れても安心です。

●その他の主な新機能

キー操作のファイルへのセーブとリプレイ、処理可能行数を拡張、リジューム機能、Ctrlキー・アサイン（オプション）、メニュー（オプション）etc。

アメリカでも日本でも高い評価

●プログラミング向きエディターを探している人は、もう探す必要はない。(Jerry Pournelle BYTE)
●ソフト開発には強力なツール、300回のアンドゥは大きな特徴。(日経バイト)
●既存のエディターに不満を覚えたら、これしかない。(Oh! FM)
●まるで夢のようなアンドゥだ――おもわず間違うのを期待してしまいそうだ。(The C Users Journal)

■主な機能

◆55ウィンドウ◆無限のファイル◆自動コンパイル◆自動エラー・ジャンプ◆自動インデント◆自動テンプレート◆バッファ書出 切換 削除 リスト表示◆ファイル名の指定 表示 変更◆バックアップ保存◆特殊文字入力◆2行を1行に連結◆ファイル単位の挿入◆増分検索◆各種マーク 始・終点の入換◆ブロックのカット コピー ペースト 印刷 保存 左右へのシフト◆対応する括弧へ移動◆C言語のコメント化 非コメント化◆DOSコマンドの実行◆キー操作記憶 実行◆入力ヒストリー◆英文テキスト整形◆タブのスペース挿入◆拡張子毎に異なった設定ができるタブ◆検索置換パターンの画面からの入力◆英字の大小文字変換 その他

定価48,000円

（消費税は含まず）

■5機種対応パッケージ

（東芝、NEC、AX協議会、IBM、富士通）

●5 2DD、3.5 2DDデュアルメディア●MS-DOS 3.1以降に対応●英語DOSでは使用できませんが、英語版BRIEFもセット販売いたしております。●お急ぎの方、上記以外のメディア希望の方、英語BRIEFのセット販売希望の方はユーザー・サポート係までご連絡下さい。

日本システム株式会社

●お問合せは　東京都立川市曙町1-18-2一清ビル立川事業所Briefユーザー・サポート係
TEL0425-23-4469　FAX0425-27-6127

〈資料請求番号004〉

新連載!!
プログラマの妻たち

12月8日発売

月刊 Cジャーナル
米国「The C Users Journal」「TECH Specialist」提携誌
C言語を主体とするコンピュータプログラマのための月刊情報誌

C JOURNAL

C JOURNALは、毎月8日全国の書店でお求めになれます。定価980円（本体951円）

1
JAN.1992

特集
移植——海外ソフトのポーティング例
■PC-9801へのゲームの移植テクニック
■GNU EmacsのMS-DOSへの移植
■国産機でMINIX386

小特集
▶HyperCardの新手法

特別記事
▶D.ThielenのMemorandum
▶Cまっぷ製作記

連載
●STANDARD C
●Dr. C's Pointers
●Questions & Answers
●Programming Style第6回
●MINIX通信 第12回
●GUI環境 第6回
●BART EXPRESS 第17回
●たのむぜ！DOS/V！ その⑥

付録 DISK
◘Cまっぷサンプル版［開発支援ソフト］
◘GNU Emacs［DOS Extender対応版］
◘掲載記事の全コード収録

次号特集（2月号／1月8日発売）
デバッギング

株式会社ビレッジセンター 出版局
〒101 東京都千代田区三崎町3－1－16 神田アメレックスビル2F
●営業部：03-3221-1825●編集部：03-3239-1950

あのVisual Basicに、待望の解説書がいよいよ登場！

98シリーズで使う 本文2色刷

Visual Basic入門

12月18日発売

A5判 160頁 定価 980円（本体 951円） 塚本吉彦著

本書の内容

第1章 Visual Basicとは
[1] Visual Basicとはどんな言語だろう
[2] PC 9800シリーズで使うには
[3] Visual Basicの画面構成を覚えよう
[4] Visual Basicの操作を覚えよう
第2章 WindowsとVisual Basic
[1] Windowsのしくみ
[2] プログラムの処理
第3章 Visual Basicの約束ごと
[1] プログラミングの基本
[2] 変数・定数・演算子
[3] 制御構造
第4章 関数・ステートメント
[1] 配列の操作
[2] 型の宣言
[3] 画面の描画
[4] プリンタへの出力
[5] 文字列の出力に関する関数
[6] ファイルの操作
■付録1 プログラム例
(1) 遡って数値を確認できる足し算専用電卓／(2) 見積・納品・請求書作成プログラム／(3) DDEの利用例
■付録2 用語解説一覧

ご注文・お買い求めは、
お近くの書店まで

発売：**株式会社ビレッジセンター 出版局**
〒101 東京都千代田区三崎町3－1－16 神田アメレックスビル2F
●営業部：03-3221-1825●編集部：03-3239-1950●FAX：03-3221-1768

次代のアドバンテージ、最新の技術をあなたに!

Microsoft® University

独学より大学。

マルチメディアのプラットフォームMS-Windows、高信頼性のMS OS/2。本格的なマルチタスクの実現、グラフィカルなユーザ・インタフェース(GUI)、LANマネージャによる高度なネットワーク処理など、技術者の創造力を刺激する環境を提供します。マイクロソフト・ユニバーシティは、その登竜門。ウインドウ環境、デバイスドライバの開発、ネットワーク処理など、アプリケーション・ソフト開発に欠かせない、ノウハウをレクチャします。入門から専門まで、充実したコースラインアップ。短時間で即開発に結びつく新しいコンピューティングスタイルを身につけて下さい。

Microsoft Windows Programming Environment ウインドウズ・プログラミング・エンバイラメント

■期間:5日間 ■受講料:18万円(税別)

■MS Windows Ver3.0のアプリケーションを作成する方のためのコースです。Windows プログラミングのポイントを重点的に解説、実習中心のコースです。

Microsoft Windows Applications Programming ウインドウズ・アプリケーションズ・プログラミング

■期間:5日間 ■受講料:18万円(税別)

■MS Windows Ver3.0対応のアプリケーションを開発するプログラマや設計者のためのコースです。ユーザインタフェースやアプリケーションなど設計ポイントがマスタできます。

Microsoft LAN Manager Programming Environment LANマネージャ・プログラミング・エンバイラメント

■期間:4日間 ■受講料:15万円(税別)

■マイクロソフトLANマネージャ(LM)用アプリケーションを開発する設計者のためのコースです。LMのアーキテクチャと特徴について解説します。

Microsoft LAN Manager Applications Programming LANマネージャ・アプリケーションズ・プログラミング

■期間:5日間 ■受講料:18万円(税別)

■マイクロソフトLANマネージャ(LM)上で動作するアプリケーションの作成や移植するプログラマのためのコースです。

MS OS/2 Presentation Manager Programming Environment プレゼンテーション・マネージャ・プログラミング・エンバイラメント

■期間:5日間 ■受講料:18万円(税別)

■MS OS/2プレゼンテーション・マネージャのアプリケーションを作成するための入門コースです。作成に必要となる基本事項を説明いたします。

MS OS/2 Presentation Manager Applications Programming プレゼンテーション・マネージャ・アプリケーションズ・プログラミング

■期間:5日間 ■受講料:18万円(税別)

■MS OS/2 PMのアプリケーションを作成するためのステップアップコースです。経験者の方を対象に、より詳しく説明します。

MS OS/2 Programming Environment プログラミング・エンバイラメント

■期間:4日間 ■受講料:15万円(税別)

■MS OS/2のアプリケーションを作成するために必要となる基本的技術を修得するコースです。本コースは、OS/2に取り組む上での基本コースとなります。

MS OS/2 Applications Programming アプリケーションズ・プログラミング

■期間:5日間 ■受講料:18万円(税別)

■MS OS/2のAPIを使用し、マルチタスク、メモリ管理、プロセス間通信、ダイナミックリンクなどの機能を有効に活用する、実習を中心としたコースです。

MS OS/2 Device Drivers デバイス・ドライバ

■期間:4日間 ■受講料:15万円(税別)

■MS OS/2のデバイス・ドライバの機能、構造を修得し、実際にデバイス・ドライバを作成するコースです。デバイスヘルパールーチンなども解説します。

C言語入門

■期間:2日間 ■受講料:6万円(税別)

■これからC言語を始めようとする方を対象に、C言語の基本的な文法からわかりやすく解説します 統合環境を持つQuick Cを使って理解を深めます

C言語応用

■期間:3日間 ■受講料:9万円(税別)

■関数、ポインタ、構造体についてプログラムを作成しながら詳しく説明します セミナは、最新のMicrosoft C Ver 6.0を使用して実施いたします

●お問い合せは

富士ソフトウエア株式会社 MU教育センタ

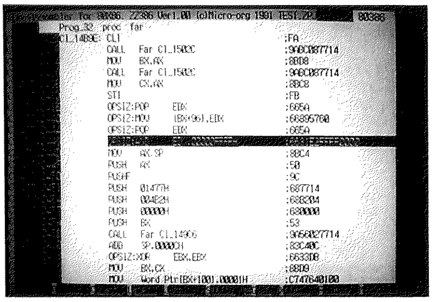
TEL.03-5461-8296 FAX.03-5461-8469

日本電気株式会社殿の御協力によりPC-9800シリーズ対応コースも好評開催中!!

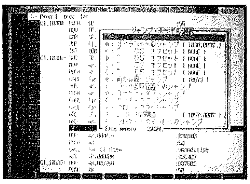
Microsoft®
マイクロソフト株式会社
〒160 東京都新宿区西新宿7-5-25 Kビルディング

富士ソフトウエア株式会社
MU教育センタ
〒108 東京都港区港南2-12-27イケダヤ品川ビル

〈資料請求番号005〉



ZZ386 最強リバース・ツール



ジャンプ

絶　　　対　　　値：指定されたアドレスへ。
実 効 オ ペ ラ ン ド：オペランドの実効的なアドレスへ。
各セグメントオフセット：オペランドの値＋セグメント値へ。
ラ ベ ル 参 照 位 置：カーソル行のラベルの参照位置へ。
各タグへのジャンプ：任意のユーザ、サーチ、クロスリファレンスのタグへ。
CS IP の 初 期 値：プログラムのスタート位置へ。
リロケートポイント：任意のリロレーションデータの行へ(EXEファイル)。
バ ッ ク ジ ャ ン プ：ジャンプのヒストリーを逆にジャンプします。

サーチ

ジャンプアドレス：指定されたアドレスへのJMP、CALL命令を探します。
ダイレクトアドレス：指定されたメモリーアドレスのMOV命令などを探します。
数　　　　　　　値：指定された数値を操作している行を探します。
特　定　の　命　令：FAR、NEAR、メモリー参照ジャンプ、I/O、INTなど特定の命令行を探します。
文　字　列　関　連：文字列の行、マッチする文字列、マッチするデータ列を探します。
セ グ メ ン ト 関 連：全ての、または指定されたセグメントレジスタを変更している行を探します。

クロスリファレンス

ラベルデータをもとにクロスリファレンスを自動作成します。同時に未定義ラベル、未使用ラベルのデータも作成されます。これらのデータはいつでも画面上で確認でき、解析のヒントになるでしょう。またジャンプ機能から参照されます。ファイル、プリンターに出力できます。

マニュアルモードの操作性

ラベルはZZ386が自動生成する以外にユーザーが任意の名前で作成、リネームできます。コメントはZZ386が自動生成する以外にユーザーが書き込めます。命令、データ(バイト、ワード、DUPなど)、文字列などの属性はスクリーン上から行、ブロック単位で指定変更できます。

その他の機能

プロジェクトファイル（逆アセンブル諸条件、動作環境の設定）のメニュー作成。テキストファイルの表示、ファイルのダンプ表示。子プロセスの実行。EXEヘッダの表示。

□強力になった自動解析能力とスクリーンエディター並みの操作で最高の環境を提供いたします。
□EMS、GVRAM、ディスクスワップ、メモリーデータのガベージにより大容量ファイルに対応。
□豊富なサーチ、ジャンプ機能は逆アセンブルを強力に支援いたします。
□ラベル、コメント、プロシジャ、セグメントが自動作成されます。
□逆アセンブルにより得られたソースファイルの全部または1部をファイル、プリンターに出力できます。
□各コマンドの使い方はオンラインヘルプにより適時に確認することができます。
□作業途中のデータはファイルに記憶され、そこから再開できます。

対応CPU：8086〜80486
動作機種：PC9801シリーズ、
PC286/386シリーズ
IBM PC/AT.DOS/V J4.0

（5"/3.5"デュアルメディア）
価格（税別）
¥39,000

ビレッジセンター、T-ZONE、システムロード、その他ソフトショップにて好評発売中です。
直接購入ご希望の方は銀行振込みまたは現金書留にてお願いいたします。送料は当社で負担いたします。郵パックの代引でも御注文承っております。
●振込口座：太陽神戸三井銀行　神宮前支店　普通 086-3056895

micro-org
株式会社 マイクロ・オーグ
〒164 東京都中野区南台1丁目1番14号サンハイツ208
PHONE:03-3375-3306 FAX:03-3373-6605
営業時間:月〜金 AM11:00〜PM6:00

※広告に記載されている各製品の小数点以下のマイナーバージョンアップについては、無償で行なっています
ご希望の方は、返信用封筒と175円切手2枚を同封の上、お申し込み下さい
（現在の最新バージョンは、ZZ386＝1.09、ZASM＝3.09、ZZ＝2.06です）

〈資料請求番号006〉

本格派Cプログラマを目指す、すべての初心者へ!!

MS-C入門

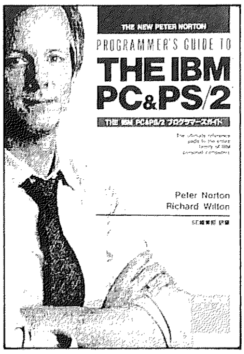
●定価2,600円　A5判・380頁
吉川敏則　著

本書は、初心者を対象にしたものですが、
初心者が初級のままとどまるのではなく、
中級のレベルに到達することを目標としています。
これまでC言語の入り口で挫折していた人も、
詳細に書かれたこの一冊を手に入れることにより、
Cプログラマへの道が開かれます。

SE Shoeisha Co., Ltd. 株式会社翔泳社 〒102東京都千代田区平河町2-4-14平河町KSビル TEL.03-3263-0447(代表)
翔泳社の本は全国どの書店でもお求めになれます。店頭にない場合は「至急」とお申し出の上ご注文いただければ10日位でお届けできます。

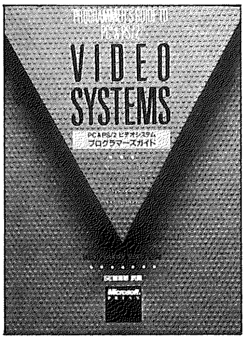
プログラミングのための優良図書

IBM PC & PS/2 プログラミング3部作



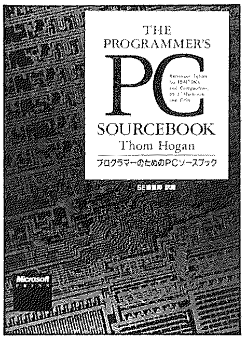
THE IBM PC & PS／2 プログラマーズガイド

●Peter Norton & Richard Wilton著
●定価5,200円
あのピーター・ノートンのベストセラー。IBMプログラマー必携のスタンダードガイガンス。



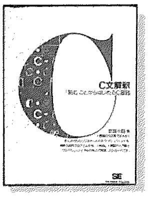
PC & PS／2 ビデオシステム プログラマーズガイド

●Richard Wilton著
●定価6,000円
IBM PCファミリーのビデオシステム完全ガイド。



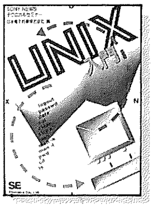
プログラマーのための PCソースブック

●Thom Hogan著
●定価5,200円
IBM PC&PS　2プログラマ必携の技術資料完訳。



C文解釈

●豊国永健著
●定価2,000円
C言語の文法書ではあまり取り上げられてこなかったわかりづらいポイントを、実際に応用プログラムを用いて解説。



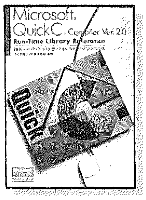
UNIX入門

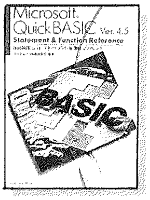
●日本電子計算株編
●定価1,800円
SONY NEWSのUNIX講座に基づくUNIXの入門書。

マクロアセンブラ プログラマーズテキスト

●吉川敏則著
●定価2,600円

最低限必要なハードとソフトの知識、データ演算の仕組みを踏まえた上でアセンブラプログラミングの基本を丁寧に解説。





Quick C コンパイラVer.2.0 ランタイムライブラリリファレンス

Quick BASIC Ver.4.5 ステートメント＆関数リファレンス

●マイクロソフト株式会社 監修
●定価各2,400円

マイクロソフトの超人気言語ソフトQuickシリーズ。Quick BASIC Ver.4.5、Quick C Ver.2.0スタンダードリファレンス。マイクロソフト社監修でオンラインヘルプを完全書籍化。サンプルプログラム満載のリファレンス。

SE Shoeisha Co., Ltd. 株式会社翔泳社 〒102東京都千代田区平河町2−4−14平河町KSビル TEL.03−3263−0447（代表）

翔泳社の本は全国どの書店でもお求めになれます。店頭にない場合は「至急」とお申し出の上ご注文いただければ10日位でお届けできます。

Programmer's Page
MONTHLY MAGAZINE FOR PROGRAMMERS

特集
プログラミングツール大集合
そのプロフィールと使いこなしのコツ

プログラミングツールとは何か？ 2
開発の工程と道具としてのプログラム 秋山登

エディット，コンパイルからドキュメント作成まで 14
フリーソフトウェアに見るMS-DOSのプログラミングツール 村木正和

プログラムをつくるための道具の系譜 28
ソフトウェア・ツール，UNIX，MS-DOS 祐安重夫

ツールの達人 34
ツールに甘えず，使いこなす 平田祥司

マクロの達人
MIFES――「外部」と「英字」，2種類のマクロをもつ 志村拓 38
VZエディタ――制御構文をもつ強力なマクロ 大野元久 44
Brief――強力なツールをC構文で書ける 梓史朗 50

創刊1周年特別企画［アンケート］ 57
本誌に登場した37人に聞きました，あなたの使っているツールを教えて！

IC千夜一夜 第7回
初期のパソコンをリードしたCPU―MCS6502― 虎岩 登 76

プログラム「遊」工房 拾弐乃巻
スクリーンセーバ 谷口典生 80

イカすBASICプログラミング入門 第1回
構造化がカッコイイ 酒井法雄 85

実践C塾 アルゴリズム篇 第1回
ソート（並べ替え）・その1 （株）ジャステックラボ 加野島英渡／山本博昭 91

98活用アセンブラ技法 SCSIドライバ編 第1回
SCSIデバイスドライバ／スケルトンの作成 星野操 103

付録
5” 2HD フロッピーディスク

綴じ込み付録
ANSI C 関数一覧

表紙イラスト：日暮修一
表紙AD：佐々木敏夫(SE)

パソコンプログラマのためのクリエイティブマガジン
月刊プログラマーズ・ページ

パソコン博物館　第13回
十進法のパソコン／カシオ計算機　FP1100　井戸禎光　72
プログラマの道具学　第13回
エディタ　祐安重夫　74
情報処理技術者試験　第２種合格／トレーニング
第10回　CASL　加藤忠宏・藤波正　113
Book Review「プログラミング WINDOWS Version 3」　中村正三郎　118
本のアンテナ　古田島義和
佐々木哲のニューヨークシステム通信　120
西川和久の希望的観測　122
コンピュータの法律　熊谷信太郎　124
アキバ短信
プログラマの仕事場　第10回　株式会社 LIFEBOAT　126

INFORMATION PAGE　129
NEWS & NEW PRODUCTS　130
Programmer's Program　TrueBASIC 用ライブラリ　134
オンラインソフト採集帳13「MIEL」　136
HEALTHY? UNHEALTHY?
ドラゴン・ポスト　138
創刊１周年記念読者プレゼント　140
付録ディスクのご案内　142
クイズ・ランダムスクリーン　143
オンラインサービス（SE-NET）のお知らせ

月刊プログラマーズ・ページ年間総目次（1991年１月号～1991年12月号）　144

広告索引　資料請求番号

(株)クレオ　P02
(株)ケーエスピー　001
ボーランド(株)　002
吉正電子(株)　003
日本システム(株)　004
ビレッジセンター
富士ソフトウエア(株)　005
(株)マイクロ・オーグ　006
T・ZONE　007
(株)ライフボート　P03
ボーランド(株)　P04

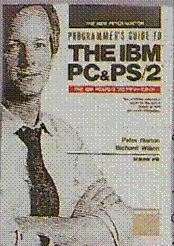
人材募集広告

(株)エルテックス
(有)美光システムワークス
(掲載順)

1992 JANUARY No.13

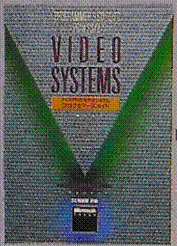
翔泳社が贈るプログラマ必携書

IBM PC & PS/2 プログラミング 3部作



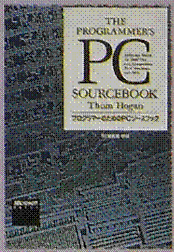
あのピーター・ノートンのベストセラー。IBMプログラマ必携のスタンダードガイダンス。

IBM PC & PS/2
プログラマーズガイド
● Peter Norton & Richard Wilton著
●定価　5,200円



IBM PCファミリーのビデオシステム完全ガイド。

PC & PS/2 ビデオシステム
プログラマーズガイド
● Richard Wilton著
●定価　6,000円



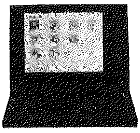
IBM PC & PS/2 プログラマ必携の技術資料宝典。

プログラマーのための
PC ソースブック
● Thom Hogan著
●定価　5,200円

IBM Authorized Dealer

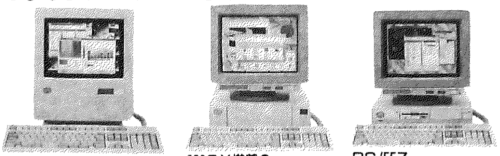
NEWマシン ぞくぞく大量入荷!!

PS/55 note



大容量80MBハードディスク内蔵 (S18 SV8 / S28 SA8)
パワフルなDOS/Vノート
80386SX(16MHz)2MB FDD内蔵

PS/55Z 30U　PS/55Z 10S　PS 55Z 10T



PS/55Z 30U XGA採用パーソナルディスクトップ
80386SX(20MHz)2MB or 6MB
パソコンらしくないパソコン? オシャレ

PS/55Z 10S 386SX搭載のスタンダードPC
DOSV, VGA, ATバス採用
80386SX(20MHz)2MB or 6MB

PS 55Z 10T PS/55Z 最強パフォーマンスPC
XGA高解像度・大容量
80386(20MHz)2MB or 6MB

NEWS とうとう登場!

日本語スーパーVGAドライバー

噂の800×600ドライバがとうとう出た
今すぐT・ZONEへ行こう!

800×600ドットのSuper VGA(S-VGA)を日本語対応にするスクリーンドライバ。
T・ZONEならでは **大特価 ¥2,500**

80486/33MHz made in USA
超特価¥498,000

T・ZONE

80486/33MHz	
RAM 8MB(max 32MB)	2シリアル、1パラレル、1ゲーム
128KBキャッシュ(max 256KB)	拡張スロット×8
3.5"1.44MB×1/5"1.2MB×1	ドライブベイ×6
200MB IDE HDD	ミニタワー、2ボタンマウス
スーパーVGA ET-4000 W/1MB	101キーボード

80386/33MHz ノート made in TAIWAN
限定20台 **超特価¥438,000**　最強ノート

80386/33MHz	外部VGA出力
RAM 2MB(max 16MB)	1シリアル、1パラレル
32KBキャッシュ	外部テンキー付属
3.5"1.44MB×1	外部101キーボード、外部FDDポート
40MB HDD	ニッカドバッテリー(充電2時間、2時間使用)
32階調VGAバックライト液晶	280W×220D×54H 4kg

80386/33MHz セット
超特価¥398,000

RAM 4MB	2シリアル、1パラレル
64KBキャッシュ	1ゲーム
3.5"×1、5"×1	マウス
80MB HDD	101キー
スーパーVGA 1MB SVGAモニター	

Microsoft BALLPOINT MOUSE **超特価¥28,000**

VGA信号をビデオに変換するコンバーター **超特価¥98,000**
320×200～640×480の信号をS-VHS、ビデオに変換して、テレビやビデオに接続できます。

CREATIVELABS
マルチメディア対応のSOUND BLASTERは必需品です! 特価提供中
SOUND BLASTER

これ1つでマルチメディアが楽しめます。
内蔵CD-ROM
SOUND BLASTER PRO
MIDI KIT
マルチメディアイクステンション他ソフト付
UPGRADE KIT ¥164,800

ステレオ22FM音源
CD-ROMインターフェイス他
SOUND BLASTER互換
Multimedia Sound Standard
PRO ¥57,800

11FM音源
MIDIジョイスティックパワーアンプマイク入力 各種ソフト付
V2.0 ¥34,800

PS/55等のマイクロチャンネル版です。
MCA ¥64,800

MINI-SCSI **超特価¥29,800**
プリンタポートにつなぐだけでSCSIデバイスが使えるようになります。
100MB HDDセット **超特価¥138,000**

Mitac 3060D
106キーセット **超特価¥270,000**
101キーセット **超特価¥258,000**

共通仕様
80386SX/16MHz、3.5"1.44MB×1、HD 40MB、スーパーVGA(512KB)、スーパーVGAモニター

GAME
時代はゲーム!? TERADRIVEや新作のソフトが山盛りでご来店をお待ちいたしております。

WALT DISNEYの永遠のアイドル、ミッキーマウスがあなたにしゃべりかけます。

MICKEY'S ABC	¥10,800
MICKEY'S 123	¥10,800
MICKEY'S ABC & SOUND SOURCE	¥13,800
SOUND SOURCE(驚異のローコスト音源スピーカーです)	¥6,800
JOYSTICK	¥6,800
ゲームカード(PC/AT)	¥9,800
ゲームカード(MCA)	¥15,000

NOTE
ノートといえば、PS/55 note 大特価提供中!

IBM	PS/5523SJ4	超特価¥198,000
SANYO	MBC-18NVH4	~~¥355,000~~
NBCC	386NOTE	¥298,000
NBCC	386LAPTOP	¥266,000
NBCC	286NOTE	¥244,000
NBCC	286LAPTOP	¥220,000

PS/55 note

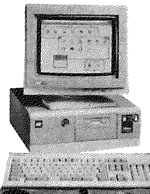
メモリー
NOTE 2MB ¥21,000
NOTE 4MB ¥40,000

DOS/Vノートおすすめセット 限定
- CPU : 80386SX/20MHz
- RAM : 3MB
- FDD : 3.5"×1
- HDD : 40MB
- 外部VGA出力
- A4ファイルサイズ

+ DOS J5.0/V + シリアルマウス + 一太郎dash

何と!何と!の **超特価¥298,000**

AST
RESEARCH JAPAN K.K. ASTから DOS/Vマシン 新登場



80386SX/20MHz
FDDモデル ¥308,000
40MBモデル ¥355,000

PCvision

VGA
DOS/Vの必須アイテムはVGAアダプターです。豊富な在庫から選べます。

BASIC VGA 640×480	¥19,800	SUPER-VGA 1024×768 256色	¥38,000
SUPER-VGA 1024×768 16色	¥38,000	Dyanbook用VGAアダプタ	¥49,800

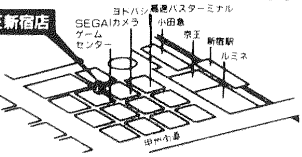
VGAモニター
DOS/V対応やWINDOWS用として高解像大画面のモニターがほしいあなた、何を選びますか?

NANAO	T660i (ケーブル別)	20"0.31mm 30～78kHz	1284×1024	~~¥588,000~~
三菱	XC-98V3	14"0.28mm 98/VGA	640×460	¥78,000
NEC	MULTISYNC II	14"0.31mm 15.5-35kHz	800×560	¥95,000
IBM	8515	14"0.28mm VGA, 8514/A, XGA		~~¥168,000~~

the Manhole (日本語版)
それはマンホールからはじまった……
ゲームを超えた
ファンタジーアドベンチャー発売中!

DOS/V対応
IBM PS/55
テラドライブ対応
DOS/V OADG
¥9,800

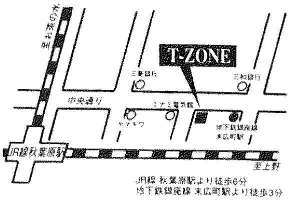
T・ZONE 新宿店 ☎03 3342-1901



1F APPLE & NEC 03-3342-1901
2F IBM PS/FACTORY 03-3342-5533
FAX 03-3342-1903
〒160 東京都新宿区西新宿1-12-18

T・ZONE PS PLAZA
IBMゾーン
3F ハードウェア
4F ソフトウェア
☎03-3257-2652
FAX 03-3257-2661

営業時間 AM10:30～PM7:00



〒101 東京都千代田区外神田4-4-1

T・ZONEスタッフ募集 IBM、Apple、無線のスタッフを募集中です。アルバイトも募集中! ☎03-3257-2690

☆T・ZONEの7Fは専門書籍(国内・外)・データーブックの専門フロアです。大量在庫中! ほしい本は何でも揃います。

〈資料請求番号007〉

PROGRAMMING TOOL

特集

プログラミングツール大集合

そのプロフィールと使いこなしのコツ

プログラミングの過程では，ソースプログラムの記述からコンパイル，リンク，テスト，デバッグ，さらにメンテナンスや改良のためにさまざまなプログラミングツールが使用されている．これらのプログラミングツールには，エディタ，コンパイラ，リンカ，デバッガなどの主要なものから，ソースジェネレータ，バージョン管理，シンタックスチェッカなど，数多くの種類がある．また，同じツールでも，コンパイラなどの言語製品に付属してくるものから単体で販売されている製品，フリーソフトウェア，シェアウェア，自作プログラムまで多様に存在している．
本特集では，プログラミングの各フェーズにおけるツールの役割を明確にするとともに，プロのプログラマが実際に使用しているプログラミングツールの概要と利用方法を紹介する．

プログラミングには，企画，設計，製造，検査，本稼働，保守といった工程が存在している．これらの各工程においてさまざまなツールが使われているが，企画や設計の段階は各社各様の形態があり，一概に述べることは困難である．また保守の中心は，製造に遡ってプログラムを修正し本稼働までの過程を通ることにほかならない．
そこで，本節ではプログラミングの範囲を本稼働を除く製造から検査までのコンピュータシステムを使用して行う範囲に限定し，ソースコードからロードモジュールを作成するまでに利用するツール，すなわちハードウェアおよびソフトウェアについて，その形態から利用方法までを解説する．

特集
プログラミング
ツール大集合

プログラミングツールとは何か？
開発の工程と道具としてのプログラム

秋山　登

PROGRAMMING
TOOL

プログラム作成の過程

　プログラムを作成する過程は，物(たとえば，自動車や電化製品など)を作る過程とほとんど同じと考えられます．図1に示すように企画に始まり，保守でその過程を終わります．プログラミングの工程とツールについて述べる前に，どのような過程を経てプログラムが作成されていくかを述べておきます．

企画

　プログラムを作成する前提としてニーズがあります．アンケートなどの調査結果や，現場の問題提起から，このようなプログラムを作成すれば商売になるとか，このようなプログラムがあれば現場の作業能率が向上するというニーズが発生します．ソフトウェアの分野では，このようなニーズのことを問題といいます．企画の段階では，この問題を解決するため問題を明確にして問題の範囲を限定し，作成するプログラムの規模や性能などを決定し，作成期間や経費などプログラム作成のための計画を立案します．企画段階で決定されたプログラムの作成に必要な事項は，要求仕様書としてプログラムの設計担当者に渡され，プログラムの設計が開始されます．

設計

　企画から設計に作業が移ると，現実のコンピュータシステムで要求仕様をどのように実現するかを検討し，入出力データ，ファイルの構造，処理のアルゴリズム，モジュール分割などについて設計が行われます．この結果が基本設計書となります．

　具体的にプログラムの製造作業を行うためには，基本設計書のままでは不十分ですから，基本設計書をさらに製造に即した形にブレークダウンします．コード部分は，フローチャート，パッド，NSチャートなどで，データ部分は変数一覧表などで製造担当者にわかるよう詳細な記述が行われます．このような記述の行われた設計書を詳細設計書と呼びます．詳細設計書は，製造担当者に渡されて，製造が開始されます．

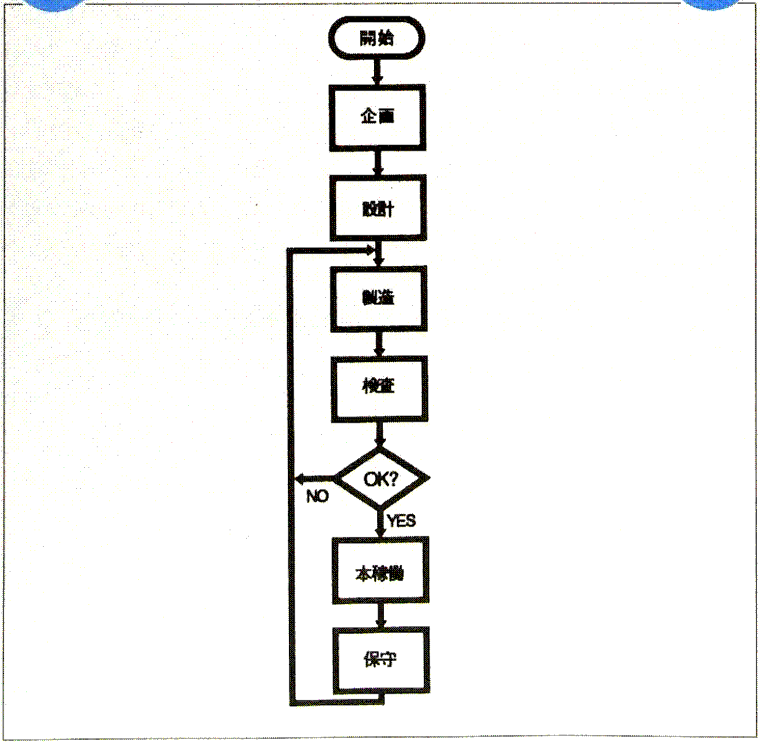
製造

　製造担当者は，詳細設計書に基づいてプログラミング言語によってコーディングを行い，ソースプログラムを作成します．そして，ソースプログラムはコンパイラなどの言語処理プログラムで処理され，最終的に実行可能なロードモジュールになります．次に検査が行われますが，検査で不合格になれば，再び製造に戻され，ソースプログラムの修正が行われます．

検査

　製造段階を経てできあがったプログラムは，最初に関数などの処理単位別，あるいはモジュール別にテストドライバを作成して，設計書通りに動作するかどうかを検査されます．検査は，設計担当者単独で，あるいは設計担当者と製造担当者が協力して行います．このような検査を単体検査と呼びます．このような検査を何段階か経て，最後にすべてのモジュールを結合した検査

FIG　図1　プログラム作成の過程　1



が行われます．このような検査を結合検査と呼びます．これらの検査で不合格になった部分は製造担当者に戻され，ソースプログラムの修正作業が行われ，その後に再検査が行われます．すべての部分が検査に合格すると，プログラムは完成したことになり，本稼働に移行します．

本稼働

完成したプログラムは，出荷されユーザーの実用に供されます．これが本稼働です．

保守

一般の機械であれば，部品が消耗して時間が経てば故障が発生します．これを未然に防ぐのが一般の機械の場合の保守です．プログラムの場合には，時間が経っても消耗することはないので，保守のニュアンスが機械の場合とは違います．

プログラムが本稼働すると，仕様書に正確に反映しにくい速度や操作性の不満がユーザーから聞かれるようになってくることがあります．また，ある特定のデータの組み合わせでのみ発生するといった検査では発見しにくいバグが発生することもあります．あるいは，プログラムの作成には人間の関与する部分が多いため，検査漏れなどの人為的ミスによるバグも発生します．

このような不都合は，プログラムの開発元の信用に関わりますから，なんらかの対処をしなければなりません．たとえば，アルゴリズムを修正したり，ソースコードを訂正して再出荷するなどの対処です．このようなプログラムの本稼働後の対処をソフトウェアの分野では保守と呼んでいます．

プログラミングのツールとしてのコンピュータシステム

プログラミングを広い意味で解釈すれば，前述の企画から保守までのプログラム作成の全段階を含めることもできます．しかし，企画や設計の段階は各社各様の形態があって一概に述べることは困難です．個人的なプログラムの作成では，頭の中で瞬間的に終わることもよくあります．また保守の中心は，製造に遡ってプログラムを修正し本稼働までの過程を通ることです．そこで，本文ではプログラミングの範囲を本稼働を除く製造から検査までのコンピュータシステムを使用して行う範囲に限定することにして，話を進めることにします．

プログラミングのツールというと，はやい話，ソースコードからロードモジュールを作成するまでに利用するものです．すなわち，コンピュータシステム全体ということになります．コンピュータシステムが，どのような構成になっているかを図2に示します．この図をもとに各要素の簡単な説明をしておきます．

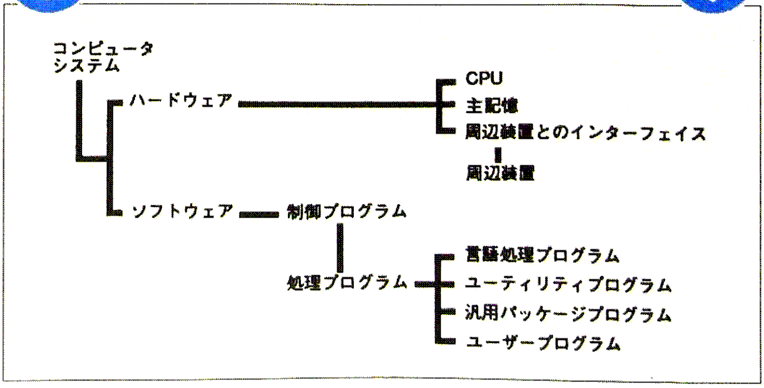
ハードウェア

ハードウェアは，CPU，主記憶，CPUと周辺装置とのインターフェースおよび，周辺装置（補助記憶装置と入出力装置の総称）から構成されます．通常のプログラミングでは，電源を入れれば動作をはじめるもの位にしか意識されません．それでも，デバッグが困難な状況に陥ったときには，CPUのかわりにICE（インサーキットエミュレータ）というハードウェアをデバッグの道具に登場させることがあります．

制御プログラム

制御プログラムは，狭い意味でのオペレーティングシステムのことです．たとえば，MS-DOSやUNIXなどがこの制御プログラムに該当します．すべてのソフトウェアはCPUにとってはまったく同列のものですが，このほかのソフトウェア，すなわち処理プログラムは，すべてこの制御プログラムのもとで動作します．以下に述べる言語処理プログラムからユーザープログラムまでを利用することは，暗黙に制

FIG 図2 コンピュータシステムの構成 2



おもなUNIXコマンド
alias コマンドエイリアスの表示・設定

御プログラムを利用することになります．

言語処理プログラム

プログラミングのツールとしては，なくてはならないプログラムです．処理の形態から，アセンブラ，コンパイラ，インタプリタ，ジェネレータ，トランスレータ，シミュレータ，およびエミュレータとそれらに付属するライブラリなどから構成されます．具体的には，C言語のコンパイラ，BASIC言語のインタプリタなどがこれにあたります．

ユーティリティプログラム

ユーティリティプログラムは，サービスプログラムとも呼ばれます．コンピュータシステムの運用やプログラム作成を容易に行えるようにするためのプログラム群から構成されます．MS-DOSの場合，COMMAND.COMに組み込まれているCOPY，DIR，TYPE，DELなどのコマンドのほか，EDLINやSYMDEBなどMS-DOS購入時に付属しているすべてのコマンドは，このユーティリティプログラムに属します．このように，ユーティリティプログラムはプログラミングにおけるツール群ということもできます．

汎用パッケージプログラム

汎用パッケージプログラムは，アプリケーションプログラムとも呼ばれます．その名の通り応用分野を限定したプログラムで，ユーザーが比較的簡単な操作を行うと結果が得られるプログラムです．プログラム作成の分野では，PERT/CPMなどの技法を使った工程管理や，流れ図の自動作成プログラムを仕様書の作成に使用することがあります．このほか，データベース管理システムやワープロのプログラムが汎用パッケージプログラムに該当します．プログラミングの立場から見れば汎用パッケージプログラムには，どちらかというと間接的な役割を果たすツールとして使用されるプログラムが多いといえます．

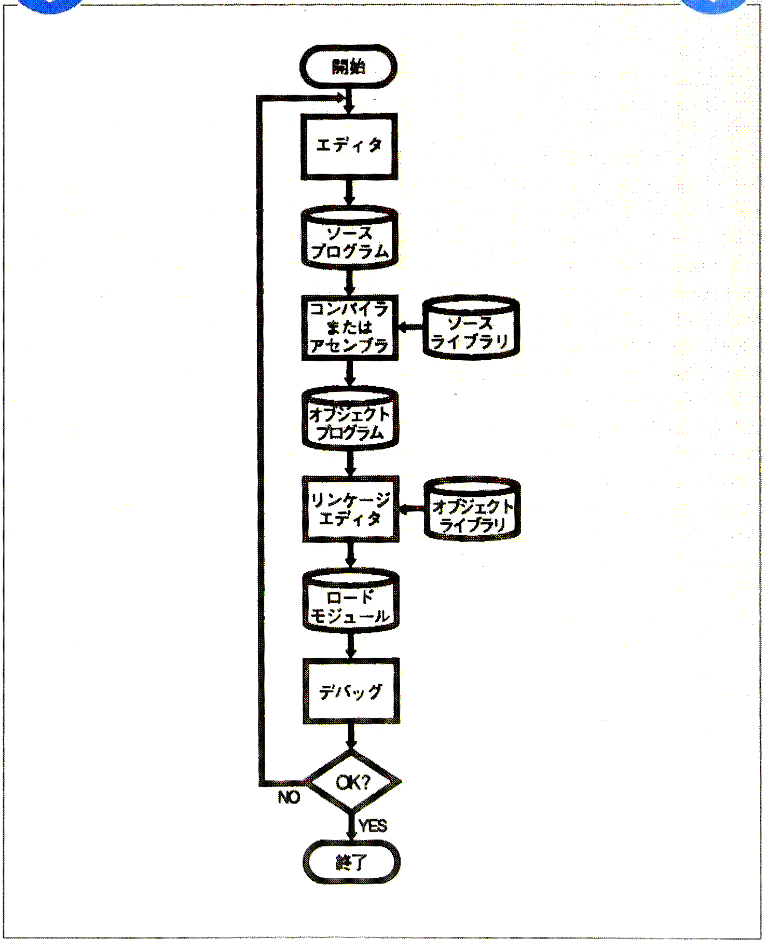
ユーザープログラム

コンピュータシステムのユーザーが，自身の業務や趣味のために独自に開発したプログラムです．銀行で稼働している巨大なオンラインリアルタイムのバンキングシステムから，趣味で作成するゲームまで多種多様なユーザープログラムが存在します．ユーザーが開発した独自のツールは，ユーザープログラムに含まれます．

プログラミングの工程

プログラミングの工程は，使用するプログラミング言語の処理系によって

FIG 図3a　コンパイラ／アセンブラを使用したプログラミング 3a



異なります．たとえば，パソコンでよく使用されるC言語のある処理系では，ソースプログラムがコンパイラによってオブジェクトプログラムに直接変換されます．別のC言語の処理系では，ソースプログラムがコンパイラによってアセンブリ言語のソースプログラムに変換され，さらにこのアセンブリ言語のソースプログラムがアセンブラによってオブジェクトプログラムに変換されます．このようなコンパイラは，1つの言語からほかの言語に変換するという意味で，トランスレータと呼ばれることがあります．どちらの場合でも，ソースプログラムはエディタを使って作成され，オブジェクトプログラムは，リンケージエディタで必要なライブラリあるいはほかのオブジェクトプログラムと結合され，ロードモジュールに変換されて実行されます．

また，BASIC言語によるプログラミングでは，ソースプログラムの作成からデバッグまでをインタプリタが受けもちます．以上述べた工程をまとめると，コンパイラによるものが図3a，トランスレータによるものが図3b，インタプリタによるものが図3cのようになります．

コーディングにおけるツール

製造はコーディングから始まります．コーディングにおけるツールの主役は，何といってもエディタです．以下にエ

FIG 図3b　トランスレータを使用したプログラミング 3b

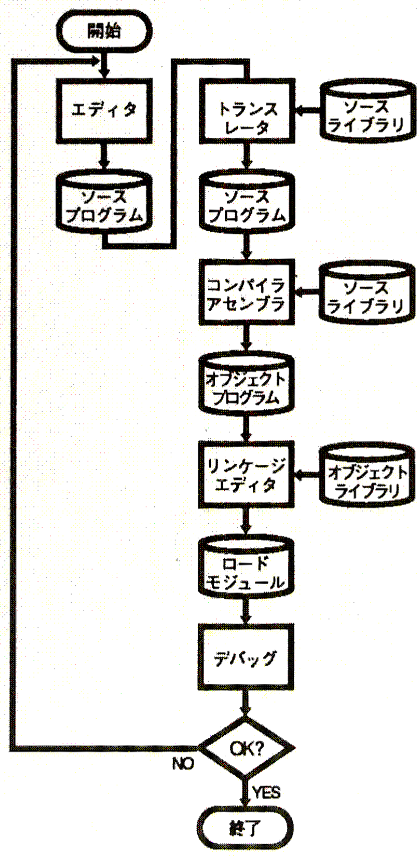
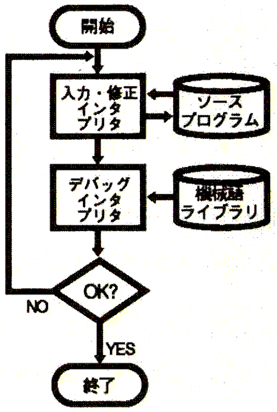


FIG 図3c　インタプリタを使用したプログラミング 3c



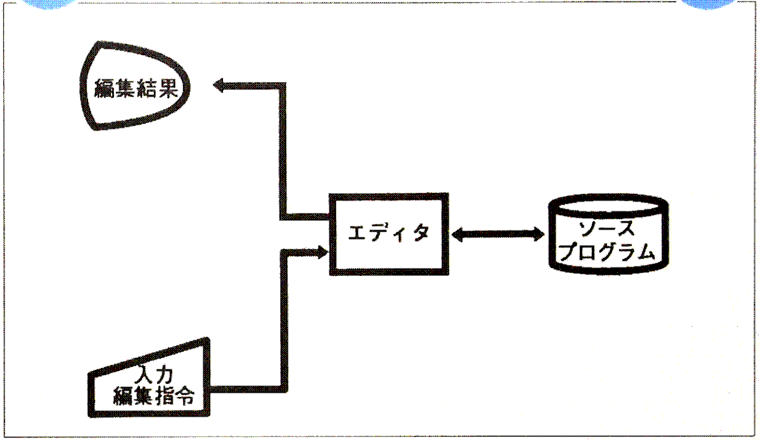
おもな UNIX コマンド
awk　レポート作成，プロトタイプ作成用言語インタプリタ

ディタを中心に，コーディングで使用されるツールを紹介します．

エディタ

エディタは，コーディングの次の段階である言語処理に必要なソースプログラムのファイルをキーボードからの入力によりディスク上に作成したり，すでにあるソースプログラムのファイルを修正したりするプログラムです．この概念を図4に示します．

FIG 図4 エディタの作業の概念 4



エディタには，ラインエディタとスクリーンエディタの2種類があります．このうち，ラインエディタは，物理的な行単位にプログラムの入力や修正を行うものです．このエディタの最大の欠点は，行単位の入力や修正を基本としているため，修正の必要な行は，その行を削除して入力し直さなければならないことです．文字列単位の置換もできますが，結果は行を表示させるコマンドを入力しないかぎり見ることのできないのも欠点の1つです．このエディタの長所は，普通の端末の最小限の機能しか使用しない点です．したがって，このエディタは使用する端末やパソコンを選びません．ラインエディタは，次に述べるスクリーンエディタが普及した今日では，過去のものになりつつありますが，低速の通信回線で接続された端末では処理速度の関係もありなお使用されています．ラインエディタの例としては，CP/M付属のEDやMS-DOS付属のEDLINがあります．

スクリーンエディタは，ソースファイルの内容をつねに画面に表示して，カーソルキーやマウスにより文字単位で任意の位置を指定して入力や修正のできるエディタです．この形式のエディタの長所は，入力や修正を失敗してもただちにその場所に移動して必要な箇所だけを修正すればよいことです．その上，最新のソースプログラムの内容がつねに表示されていますから，確認作業もラインエディタより遙かに楽になります．スクリーンエディタの欠点は，複雑な画面操作の機能を使用したり，VRAMを直接アクセスする処理を行うため，ハードウェアにあったスクリーンエディタを用意しなければならないことです．スクリーンエディタの例としては，MIFES，RED，Vzエディタなど多数の製品が市販されています．

最近のスクリーンエディタは，いろいろな機能をもっていますが，その中から主要なものを以下に列挙します．

【複数ファイルの同時編集】

複数のファイルを同時にオープンして，相互に参照したり，内容を移動したりする機能です．1つのプログラム作成で，多数のソースプログラムファイルを作成している場合に便利です．

【マルチウィンドウ】

画面を分割して，複数のソースプログラムの内容を同時に表示する機能です．複数のプログラム間の相互参照や，内容の比較に使用します．

【マクロ】

多数のキー操作を1つのキーに割り当てたり，専用のマクロ記述言語を使用して複雑な操作を記述し1から2のキーストロークに割り当てたりする機能です．複雑な編集操作を連続して行うとき，キー操作が格段に楽になります．

【カスタマイズ】

自分の好みに応じて，キーに割り当てられている編集機能を変更したり，画面の配色を変更したり，画面の配置を変更したりできる機能です．好みにあったエディタを使用することは，作業能率の向上につながります．

【タグジャンプ】

最近のアセンブラやコンパイラは，診断メッセージの先頭に原因となったソースプログラムのファイル名と行番号を表示します．たとえば，MS-Cのコンパイラは，次のようなメッセージを表示します（図5）．

この診断メッセージをリダイレクトしてファイルに格納しておき，エディ

FIG　図5　MS-Cコンパイラの診断メッセージ　5

```
TEST.C(33) : warning C4071: 'prntf' : no function prototype given
```

タにそのファイルを読み込ませます．次にそのファイルを編集するのと同様にカーソルを診断メッセージの先頭に移動してファイルと行番号を指定することにより，ソースプログラムの診断メッセージの原因となった行から編集をはじめる機能です．単純な文法ミスの修正に威力を発揮します．

【リファレンスマニュアルの参照】

参照したい単語（たとえば関数名やキーワード）の先頭にカーソルを移動してこの機能を呼び出すと，その単語に関する説明が表示される機能です．多数のライブラリを使用するとき，使用法を簡単に見つけることができます．

【言語処理プログラムの実行】

あらかじめ，言語処理プログラムのコマンドラインの書式を指定しておくと，その書式にしたがってコマンドラインを展開し，言語処理プログラムを起動する機能です．言語処理プログラム実行中は，エディタ自身の現在の状態をファイルにセーブしてメモリを言語処理プログラムに明け渡し，言語処理プログラムの実行が完了すると，エディタの実行を再開して元の状態に復帰します．

ファイル間の文字列検索プログラム

最近のプログラミングの傾向として，構造化プログラミングの普及があります．構造化プログラミングでは，モジュールを機能別に細分化して多くのソースファイルが作成されます．この結果，再利用可能なモジュールが多く生まれ，あるプログラミングで作成されたプログラミングを別のプログラミングで利用する機会がよく発生するようになります．このような場合，多くのソースファイルの中から必要とするソースファイルを素早く検索するようなツールが必要となります．同様に，運悪く多くのソースファイルの修正を伴うような修正を行わざるを得なくなった場合にも，必要とするファイルの検索を漏れなくそして素早く行う必要があります．

このような機能を果たすユーティリティプログラムがUNIXシステムで使用されているGREPです．GREPは，最近ではMS-DOS用のものもあります．また，ほとんどのスクリーンエディタにはGREPと似たような機能を果たすユーティリティプログラムが付属しています．

GREPで検索する文字列を指定するときには，正規表現と呼ばれる独特の記述方法を使用します．たとえば，[A-Z]はAからZまでの任意の英大文字に一致します．GREPに検索するファイルの範囲と，検索する文字列を指定して起動すると，検索する文字列と一致した文字列が見つかったときには，先に述べたタグジャンプのできる形式の情報と，行の内容を出力します．

ファイル管理ツール

ソースプログラムファイルを編集するとき，保存用のディレクトリから作業用のディレクトリにファイルをコピーして編集を行うプログラマが多いはずです．このとき，編集の対象にしているファイルの数が少なければあまり問題ないのですが，多くなるとどのファイルが編集済みで，どのファイルが未編集か分からなくなってしまうことがあります．この問題を解決してくれるのがファイルのアップデート管理ツールです．

ファイルのアップデート管理ツールは，保存用のディレクトリと作業用のディレクトリの間で，ディレクトリに登録されているファイルの最終更新日時のデータに基づいて動作します．作業用のディレクトリのファイルが，保存用のディレクトリに存在する同じ名前のファイルよりも更新された日時が遅ければ，作業用のディレクトリから保存用のディレクトリにファイルをコピーします．コピーの後，とくにオプションで指定されていれば，コピーされた作業用ディレクトリのファイルを削除します．

バージョン管理ツール

バージョン管理ツールは，何人かのチームで大規模な開発を行っているときによく使用されます．毎日チームで作成されたり，修正されたソースプログラムファイルは，チームの管理者の作業用ディレクトリに送られます．バージョン管理ツールは，これらのソースプログラムを管理者用の保存ディレクトリにコピーするとき使用されます．このときの動作は，次のようなものです．

バージョン管理ツールを使用して新規のソースプログラムファイルを保管ディレクトリに登録するとき，ソースファイルの先頭にコメントの形式でバージョン管理に関する情報が付加されます．次に，同じ名前のファイルを登録するとき，前に登録されたファイルのバージョン管理の情報から新しいバージョンの情報が作成され，新しいファイルにその情報が付加されて登録されます．バージョン管理ツールには，ファイルの登録に関する情報をファイルに保存したり，世代保存を行うものもあります．

言語処理とリンクのツール

言語処理とリンクの役割は，ソースプログラムを実行可能なロードモジュールに変換することです．ただし，BASIC言語のようにインタプリタで処理されるものは例外です．この段階で使用される主要なツールは次のとおりです．

プリプロセッサ

ソースプログラムを作成する際に，ソースの記述を容易にしたり，ソースを読みやすくするために直接言語処理プログラムで処理できない記述をすることがあります．たとえば以前にはよくあったことですが，全角文字の処理できないコンパイラで処理するソースプログラムの中に全角文字で文字列を記述する場合です．このような場合には，ソースプログラムを直接コンパイラで処理することができませんから，プリプロセッサと呼ばれるプログラムでコンパイラが処理できるようにソースプログラムを変換して，変換後のソースプログラムをコンパイラに入力します．

アセンブラやコンパイラと一体になったプリプロセッサもあります．たとえば，C言語の#includeや#defineなどのプリプロセッサ文を処理する機能や，アセンブラのマクロ機能などがこれにあたります．これらの機能は，本来の言語が処理される以前に働き，ソースプログラムを本来の言語だけで構成されるように内部的に修正します．

アセンブラ

アセンブリ言語で書かれたソースプログラムをオブジェクトプログラムに変換します．アセンブリ言語は，CPUが直接実行する機械語と1対1に対応していますので，プログラムを実行するCPUによってまったく異なります．また，同じCPUでもアセンブラが異なれば，アセンブリ言語が多少異なることがあります．

アセンブラは，機能によって分類されています．ソースプログラムで指定された主記憶の物理アドレスにしたがって直接機械語を生成するアセンブラを絶対形式のアセンブラと呼びます．絶対形式のアセンブラは，ソースプログラムで指定されたアドレス以外では実行できないので，CP/Mのように実行開始アドレスがつねに決まっているようなシステムやROM化を行うようなプログラムで利用されてきました．

絶対形式のアセンブラに対して，MS-DOSやUNIXのように実行が開始されるメモリの物理アドレスが，実行時までわからないようなシステムでは，リロケータブルアセンブラが使用されます．リロケータブルアセンブラは，プログラムで最初に割り当てられるメモリのアドレスを仮に0番地として機械語を生成し，生成した機械語とともにロードされる主記憶のアドレ

LIST 1 リスト1 EX01.ASM

```
MSDOS   MACRO   FUNC
        MOV     AH,FUNC
        INT     21H
        ENDM
;
        PUBLIC  EX
TEXT    SEGMENT
        ASSUME  CS:TEXT,DS:TEXT
;
EX:     MOV     AX,CS
        MOV     DS,AX
        MOV     DX,OFFSET MSG
        MSDOS   09H
        XOR     AL,AL
        MSDOS   4CH
MSG     DB      'This is an example.$'
;
TEXT    ENDS
;
STAK    SEGMENT STACK
        DB      256 DUP(?)
STAK    ENDS
;
        END     EX
```

LIST　リスト2　EX01.ASMのアセンブリリスト　2

```
                                  MSDOS   MACRO   FUNC
                                          MOV     AH,FUNC
                                          INT     21H
                                          ENDM
                                  ;
                                          PUBLIC  EX
0000                              TEXT    SEGMENT
                                          ASSUME  CS:TEXT,DS:TEXT
                                  ;
0000  8C C8                       EX:     MOV     AX,CS
0002  8E D8                               MOV     DS,AX
0004  BA 0011 R                           MOV     DX,OFFSET MSG
                                          MSDOS   09H
0007  B4 09                   1           MOV     AH,09H
0009  CD 21                   1           INT     21H
000B  32 C0                               XOR     AL,AL
                                          MSDOS   4CH
000D  B4 4C                   1           MOV     AH,4CH
000F  CD 21                   1           INT     21H
0011  54 68 69 73 20 69           MSG     DB      'This is an example.$'
      73 20 61 6E 20 65
      78 61 6D 70 6C 65
      2E 24
                                  ;
0025                              TEXT    ENDS
                                  ;
0000                              STAK    SEGMENT STACK
0000  0100[                               DB      256 DUP(?)
        ??
                         ]
0100                              STAK    ENDS
                                  ;
                                          END     EX
```

スによって機械語の変更の必要な箇所に印を付けた特別な形式のファイルを出力します．これがオブジェクトプログラムです．

絶対形式とリロケータブルといった分類のほかにもいくつかの分類があります．マクロ機能の付いたアセンブラをマクロアセンブラと呼びます．アセンブラを実行するCPUとアセンブルされるプログラムを実行するCPUが異なるようなアセンブラをクロスアセンブラと呼びます．MS-DOSで使用されるアセンブラであるMASMは，リロケータブルアセンブラでありマクロアセンブラでもあります．

リスト1は，MS-DOSのMASMアセンブリ言語によるソースプログラムの例です．このプログラムは，「This is an example.」というメッセージを画面に表示します．リスト2は，このソースプログラムをMASMで処理した結果できあがるソースプログラムと機械語の対応を示すアセンブリリストの一部です．アセンブリリストのいちばん左側にある4桁の16進数がプログラムの先頭からの相対的な主記憶の番地を表しています．そして，その右側にある一連の16進数がソースプログラムに対応する機械語です．

コンパイラ

コンパイラは，リロケータブルアセンブラと同じようにソースプログラムを機械語に変換してオブジェクトプログラムを生成します．コンパイラがアセンブラと大きく異なる点は，処理するソースプログラムの内容がアセンブラのように機械語と直結したものではなく，数式のように機械語とは無関係な人間が日常使用する記述（高級言語という）に近いものである点です．

コンパイラで処理されるプログラミング言語には，C，FORTRAN，Pascal，COBOL，PL/Iなどがあります．リスト3は，C言語によるソースプログラムの例です．このプログラムは，1から1000までの数を加算してその結果を画面に表示します．リスト4は，C言語のコンパイラがこのプログラムをオブジェクトプログラムに変換した様子を等価なアセンブリ言語で示すものです．

トランスレータ

8086CPUが出現した当時，8080CPUのアセンブリ言語のソースプロ

LIST　リスト3　EX02.C　3

```
#include <STDIO.H>

void main(void) {
        int     i;
        long    x;

        x=0L;
        for(i=1; i<1000; i++) x+=(long)i;
        printf("sum = %ld¥n", x);
}
```

LIST 4　リスト4　EX02.Cのオブジェクトプログラムをアセンブリ言語に変換したプログラム

```
                TITLE           EX02.C
                .286p
                .287
                INCLUDELIB      LLIBCE
EX01_TEXT       SEGMENT         WORD PUBLIC 'CODE'
EX01_TEXT       ENDS
_DATA           SEGMENT         WORD PUBLIC 'DATA'
_DATA           ENDS
CONST           SEGMENT         WORD PUBLIC 'CONST'
CONST           ENDS
_BSS            SEGMENT         WORD PUBLIC 'BSS'
_BSS            ENDS
DGROUP          GROUP           CONST, _BSS, _DATA
                ASSUME          DS:DGROUP, SS:DGROUP
                EXTRN           __acrtused:ABS
                EXTRN           _printf:FAR
;
_DATA           SEGMENT
$SG170          DB              'sum = %ld', 0aH, 00H
_DATA           ENDS
;
EX01_TEXT       SEGMENT
                ASSUME          CS:EX01_TEXT
                PUBLIC          _main
_main           PROC            FAR
                ENTER           4,0
                SUB             AX,AX
                MOV             WORD PTR [BP-2],AX
                MOV             WORD PTR [BP-4],AX      ;x
                MOV             BX,1
$F167:          MOV             AX,BX
                CWD
                ADD             WORD PTR [BP-4],AX      ;x
                ADC             WORD PTR [BP-2],DX
                INC             BX
                CMP             BX,1000
                JL              $F167
                PUSH            WORD PTR [BP-2]
                PUSH            WORD PTR [BP-4] ;x
                PUSH            DS
                PUSH            OFFSET DGROUP:$SG170
                CALL            FAR PTR _printf
                LEAVE
                RET
_main           ENDP
EX01_TEXT       ENDS
;
                END
```

グラムを8086CPUのアセンブリ言語のソースプログラムに変換するためのプログラムが使用されたことがありました．このようにある言語のソースプログラムから，ほかの言語のソースプログラムを作成するプログラムのことをトランスレータと呼びます．トランスレータの出力はソースプログラムですから，さらにアセンブラやコンパイラで処理する必要があります．

インタプリタ

BASICやLISPは，インタプリタと呼ばれる言語処理プログラムで処理され実行されます．インタプリタではアセンブラやコンパイラと異なりソースプログラムをCPUが直接実行できるような形式に変換することはありません．そのかわりにソースプログラムと1対1に対応した内部表現に変換するのが普通です．インタプリタでは，インタプリタがCPUにかわってソースプログラムや内部表現を1要素ごとに解釈してインタプリタの機能を使って実行します．したがって，コンパイラやアセンブラで処理して実行する形式の言語に比べると処理速度が遅くなるのが欠点です．

インタプリタは，それ自体で完結するプログラミング環境です．ソースプログラムの入力と修正，および実行を同一のインタプリタ上で行うことができます．プログラムの実行時には，インタプリタが致命的なエラーを常に監視することができますから，プログラムが暴走することがないばかりか，問題が発生すると即座に報告してくれます．このようにプログラムを開発する際の対話性がよいので，インタプリタで処理するプログラミング言語は会話型言語と呼ばれます．

インタプリタで処理される言語はたいていの場合，コンパイラで処理して処理速度を向上させることができます．ただし，コンパイラで処理すると対話性は犠牲になり，文法上の制約を受けることがあります．

リンケージエディタ

リンケージエディタは，リンカとも呼ばれ，アセンブラやコンパイラの出力するオブジェクトプログラムと，あらかじめ用意されている実行のために必要なオブジェクトプログラムの集合であるライブラリとを結合してロードモジュールを作成します．ロードモジュールはオペレーティングシステムのローダによって主記憶にロードし実行するためのプログラムファイルで，MS-DOSの場合は，.EXEファイルや.COMファイルがこれに相当します．MS-DOSで動作するリンケージエディタの例として，MS-LINK，PLINK86，RTLinkなどがあります．

リンケージエディタは，ライブラリのほかに複数のオブジェクトプログラ

ムを結合することもできます．モジュール分割されたソースプログラムは，複数のソースプログラムファイルとなります．コンパイラは，1つのソースファイルに対して1つのオブジェクトプログラムファイルを作成します．

複数のオブジェクトプログラムやライブラリとの結合は，プログラマがプログラムや変数に付けた名前やコンパイラが知っている名前をアセンブラやコンパイラがアドレスのかわりに，あるいはアドレスと対応させてオブジェクトプログラムに出力することによって行われます．リンケージエディタは，オブジェクトプログラムやライブラリに含まれている名前とアドレスの対応表を作成して，名前をアドレスに変換して機械語を完全なものにします．

リンケージエディタのもう1つの重要な役割は，デバッグのための情報を作成することです．もっとも簡単なデバッグのための情報は，リンケージエディタが処理した名前とアドレスの対応表で，マップと呼ばれます．この情報は，機械語のレベルでプログラムをデバッグするときに役に立ちます．最近では，機械語を知らなくてもソースプログラムのレベルでデバッグできるデバッガもよく使用されるようになっています．このための情報も，リンケージエディタがファイルとして作成します．この情報には，ソースファイルの中に含まれるすべての名前とそのアドレスなどの対応表，ソースファイルの行番号とアドレスの対応表などが含まれています．これらの情報の元になるデータは，オプションの指定によってアセンブラやコンパイラがオブジェクトファイルの中に書き込んで，リンケージエディタに渡します．

ライブラリアン

多数のオブジェクトプログラムをリンケージエディタで1度に処理しなければならないとき，オブジェクトプログラムを1つのファイルにまとめると，リンケージエディタの操作が楽になります．このようにしてできあがったオブジェクトプログラムの集合体をライブラリと呼びます．ライブラリアンは，ライブラリを作成したり，ライブラリから特定のオブジェクトプログラムを削除したり，新たにオブジェクトプログラムを追加したり，ライブラリの中身を調べたりする機能をもっています．

メイク

多数のソースファイルからロードモジュールを作成する場合，あるソースファイルが更新されたにもかかわらず，コンパイルとリンクの実行を忘れてデバッグの段階で混乱することがよくあります．メイクは，このようなときにロードモジュールとソースファイルの最終更新日時からコンパイルとリンクを実行するかしないかを決定して，必要であればコンパイルとリンクを実行させるバッチプログラムの処理システムに類似する機能をもつユーティリティプログラムです．

バッチプログラムに相当する部分は，メイク記述ファイルと呼ばれます．次に，1つのソースファイルが更新されていれば，コンパイルとリンクを実行するようなメイク記述のもっとも簡単な例をあげましょう(リスト5)．

リスト5の例で，ex3.exeはターゲットファイル，ex3.cは依存ファイルと呼ばれます．メイクを実行すると，ターゲットファイルと依存ファイルの最終更新日時が調べられ，もし依存ファイルのほうが新しければ，2行目に書かれているコマンドが実行されます．例では，2行目のコマンドは，MS-CのコンパイラをE起動してC言語のソースプログラムex3.cをコンパイル，リンクしてex3.exeを作成するコマンドです．もし，依存ファイルの最終更新日時がex3.exeと同じか古いならば，メイクは何も実行しないで処理を終了します．

メイクには，マクロ機能や依存ファイルからターゲットを作成する方法を推論する機能を含む数多くの機能があり，複雑なメイク記述ファイルの作成は初心者にはなかなか困難です．そこで統合開発環境には，ファイルの範囲を指定しておけば自動的にメイク記述ファイルを作成する機能がたいてい付属しています．

LIST 5　リスト5　簡単なメイク記述

```
ex3.exe:ex3.c
        cl /AL ex3.c
```


おもなUNIXコマンド
chmod　ファイルモードの変更

デバッグのツール

アセンブルやコンパイルとリンクが終わって，ロードモジュールができあがると検査のための実行が行われます．よほど簡単なプログラムでないかぎり１度のアセンブルやコンパイルでは正常に動作しません．大規模なプログラム開発では，モジュールごとにテストドライバを作成して，入念な検査が行われます．検査の過程で，プログラムが運悪く暴走して，デバッグが行き詰まることがあります．このようなときには，次に述べるデバッガが使用されます．デバッガでは，ロードモジュールをデバッガに読み込ませて，デバッガ管理のもとでプログラムを実行させます．

機械語レベルのデバッガ

機械語レベルのデバッガでは，バグの原因を発見するため，機械語の１ステップ単位に実行状態を監視することができます．実行状態の監視は，レジスタの内容や主記憶の内容をダンプしたり，実行中の機械語をアセンブリ言語に変換(逆アセンブル)したりして行います．必要であれば，レジスタの内容や主記憶の内容を設定することができます．また，機械語の内容を変更したり追加するために，簡易アセンブラを利用することもできます．

デバッガで，プログラムの最初から実行状態を追跡したのではバグの原因究明までにかなり時間がかかることがよくあります．そこで，バグの症状などから大体のバグの位置を特定しておきその周辺を重点的に調べるのが普通です．このために，デバッガにはブレークポイントを設定する機能があります．デバッガでは，指定されたアドレスからデバッグの対象となっているプログラムをデバッガの関与なしに通常の実行と同じように走らせることができますので，重点的に調べたい位置の先頭にブレークポイントを設定して，プログラムを実行させるとブレークポイントまでプログラムは通常の実行と同じに実行されます．そして，ブレークポイントでプログラムの実行は停止して，デバッガに制御が戻ります．そこで，それ以降は１ステップずつデバッガに機械語を実行させてバグの原因を究明します．

機械語レベルのデバッガでは，絶対アドレスで機械語やデータの位置を指定するのが原則ですが，中にはリンケージエディタの出力するマップの情報を利用して，名前でアドレスを指定できるものもあります．このようなデバッガをシンボリックデバッガとよびます．シンボリックデバッガの例として，MS-DOSのSYMDEBをあげることができます．

この方式のデバッガは，アセンブリ言語によるプログラミングには向くのですが，高級言語によるプログラミングには無理があります．かなり熟練しないと，機械語をソースプログラムに対応させることが困難だからです．そこで高級言語のプログラムの場合，次に述べるソースレベルのデバッガを使用するのが無難です．

ソースレベルのデバッガ

ソースレベルのデバッガは，リンケージエディタの出力するデバッグのための情報を利用して，機械語レベルのデバッガの機能にソースプログラムの１行単位でも実行状態の監視を行えるようにしたものです．したがって，ブレークポイントの設定や実行がソースプログラムの１行単位に行えます．プログラムのコード部分もアセンブリ言語とソースプログラムの両方で行うことができます．また変数の参照も，構造体のような複雑なデータ構造をもつ場合でも名前で行うことができます．

MS-DOSで使用できるソースレベルのデバッガとしては，CodeViewが有名です．最新のCodeViewでは，デバッグ情報の表示をマルチウィンドウで行いますので，実行中のソースコード，変数の状態，スタックの状態などを同時に見ることができます．

あとがき

ソースプログラムからデバッグに至るプログラミングで使用されるソフトウェアをツールとして捉え，その中の基本的なものだけを紹介しました．統合開発環境についてはとくに説明しませんでしたが，本文で述べた基本的なツールが理解できれば，エディタを中心にこれらを有機的に呼び出せるように構成したツール群として理解することは容易でしょう．多くのツールの存在を知っているということは，プログラミングの能率向上に役立ちますが，ツールの動作を掘り下げて研究してみることは，自分のプログラミング技術を磨くことにつながります．

フリーソフトウェアという概念が日本に定着してからまだ日は浅いが，すでにパソコン通信などを通して入手可能なフリーソフトウェアの数は数千本を越えている．これらのフリーソフトウェアの中には，プログラミングに利用できる便利なツールも数多く存在している．
ここではとくに，ソースコードの作成からコンパイル・リンク，バージョン管理，さらにドキュメントの作成に活用できるテキストフォーマットについて，その特徴と利用法を紹介する．

特集
プログラミングツール大集合

エディット，コンパイルからドキュメント作成まで
フリーソフトウェアに見る MS-DOSのプログラミングツール

村木正和

PROGRAMMING TOOL

ある程度，プログラミングのノウハウを得，実際にプログラムを組めるようになったとき，簡単なツールやユーティリティを組みたくなるかもしれない．そういう思いをもったとすれば，それはラッキーなことである．将来，パソコン通信を通して，あらたなる交流をもつ可能性を十二分に秘めているのである．

ところで，今回の特集はプログラミングツールである．プログラミングツールといえば，マイクロソフトのPWB，ボーランドのIDEなどの統合環境にあるツールを用いてプログラミングするということがよく話題にのぼるが，実際のところ，そのような統合環境を用いてプログラムを生産するプログラマはどのくらいいるのだろうか？

実際，筆者や筆者を取りまく人達のなかには，プログラミングでそのような統合環境を用いている人はほとんどいない．いつも$やA>に似たプロンプトが表示されるコマンドラインからソースコードの作成，オブジェクトコードの生成を行っている．MS-DOS用でアセンブリ言語で組まれたプログラムならば，拡張セットに付いてきたSYMDEBでトレースを行うことがたまにあるくらいで(筆者のMS-DOSマシンはPC-9800シリーズである)，たいていの場合は，ソースコードレベルでデバッグを行ってしまう．

実のところを話すと，筆者はプログラマが本職ではない．ただ，仕事の中でプログラムを作る機会が多いのは事実である．しかしながら，それらのプログラムを筆者あるいは筆者の所属する組織・団体の独占排他的な利益のために使おうなどとということはこれっぽっちも思わない．逆に，自分の作ったソフトウェアを必要とする多くの人達に使ってもらいたいという要望がある．であるからして，さまざまなプログラマから好意で提供されたフリーソフトウェアの恩恵にあずかりながら，そのフリーソフトウェアを使ってできたものをフリーソフトウェアとして提供していきたいと思っている．

ここでは，プログラミングがわかりかけてきた方々に向けて，プログラミングを行うにあたって必要なツールの紹介をフリーソフトウェアから探ってみる．あくまで紹介であるから，入手にあたってのishファイルからの復元，アーカイブファイルの展開(LHAに限っては解凍)についての説明は省略する．

ところで，ソフトウェアを作るにあたっては，いくつかのソフトウェアが当然のように必要になってくる．

- ソースプログラムおよびドキュメントの作成にエディタ
- 日本語入力が必要であれば日本語入力フロントエンドプロセッサ
- オブジェクトコードの作成にコンパイラ・アセンブラ・リンカ
- リビジョン管理にRCS
- ドキュメントの清書にテキストフォーマッタ

恵まれていることに，もしCでプログラムを組むようなことがあれば，これらのソフトウェアはすべてフリーソフトウェアとして入手できる．MS-DOSのシステムとフリーソフトウェアで，簡単なプログラム開発の環境が構築できてしまうのである．

ただ，「フリーソフトウェア」だからといって，ただで手に入るという安直な考えは禁物である．それを作った人の苦労をねぎらい，提供してくれたことに感謝して使うのが本筋であるということは説明するまでもないであろう．

テキストエディタいろいろ

テキストエディタの種類

一般に，データを入力し，メモリやディスクに記憶させるデータを作成・編集するのにはエディタを使う．ソースコードの入力・編集も，同じようにエディタを使うことになる(catコマンドやMS-DOSのCOPYコマンドを使ってファイルを書き込むプログラマもまれにいるらしい)．

また，日本語で表示するプログラムを作成したり，日本語でコメントを入れたり，あるいはプログラム内に書かなくても，ドキュメントを日本語で書くためには日本語入力フロントエンドプロセッサが必要となる．実際，MS-DOSマシンのユーザーなら，誰でも日本語入力フロントエンドプロセッサをすでにもっていることであろう．NECのPC-9800シリーズならNEC-AIかな漢字変換がついているし，東芝のJ-3100シリーズならATOK，エプソンの互換機ならEGBridgeが手に入っているはずである．参考までにWXPというフリーソフトウェアが(株) エー・アイ・ソフトから提供されているということも書き添えておく．

さて，本題のエディタであるが，ひとくちにエディタといってもさまざまな種類があり，通常に文書を編集するものから，原稿締切近くになると筆者がもっとも恐れるもの(？)もある．

一般にプログラムを作成する（＝ソースコードを入力する）エディタといえば，普通はテキストエディタの類を指すのだが，テキストエディタといっても，細かく分ければ種類はいろいろである．画面上での見かけから分類すると，大きく分けて次のようになるだろう．

①テキストの入力・編集を1行単位で行うラインエディタ
②テキストの入力・編集を，ウィンドウに映しだされている部分に対して行うスクリーンエディタ
③ファイルや標準入力から順々に読み出し，それを必要に応じて逐次加工していくストリームエディタ（非対話式のため，出力は処理の結果のみとなる）

①は，MS-DOSではEDLIN，UNIXではed, ex, editがこれにあてはまる．入力されたテキストの加工・編集はコマンド入力によって行われる．

②は，現在出まわっているテキストエディタのほとんどがこれにあてはまるだろう．このエディタにもいろいろな種類があり，操作体系をとってもviに準拠したもの，Emacsに準拠したもの，MifesやVzなどの市販エディタに準拠したものなどいろいろな種類がある．ほとんどのユーザーが，このスクリーンエディタを利用していることだろう．

また，③はsedに代表される．このストリームエディタは，ラベルやオペランドの変更を一気に行うとき，文書の表記統一を行いたいときなどに非常に有効である．かつて筆者は，Z80アセンブリ言語で書かれたソースコードを8086アセンブリ言語のソースコードに変換するときに，ほんのわずかではあるが助けてもらったことがある．

テキストエディタは，1つのソフトウェアを完成させるにあたってもっとも使う時間の多いソフトウェアである．それゆえに，自分にとって使いやすいものをそろえておいたほうがよいことはいうまでもない．しかしながら，数多くあるエディタの中から，自分にとってもっとも使い勝手のよいものを選ぶためには，それだけ多くのエディタを試してみなければならない．だが，将来，自分がプログラミングライフにかける時間のことを考えれば（あるいはプログラミングやドキュメントライティングにかけるトータル時間が加算されていくにつれ），その数あるソフトウェアを選ぶのに費やす時間はかなり小さな割合になっていくことであろう．もっとも，たまたま偶然に入手したソフトウェアを使い込んでいって慣れてしまうということもある．BBSのホストにアクセスし，オンラインのエディタを使って文書を書くとしたら，当然このケースになる．現在，あらゆるビジネスシーンで一太郎を使っているユーザーなどは，ほとんどがこのケースかもしれない．:-)

筆者がはじめてアセンブリ言語でプログラミングを行ったときのマシンはMZ-80Kで，使えるテキストエディタといえば，ミニコンNOVAのサブセットといわれた操作体系のラインエディタしかなかった．ちょうどCP/M（当時の8ビットマイクロコンピュー

CHART 1　表1　ダイヤモンドカーソル，vi，Emacsの基本的なキー操作

操作	vi(stevie/elvis)	WordMaster(WordStar)	Emacs(Micro Nemacs)
カーソル移動			
次の文字へ	l or space	^D	^F
次の単語へ	w or W	^F	M F
次のタブストップへ	^I	^I	^I
前の文字へ	h or Bs	^S	^B
前の単語へ	b or B	^A	M B
行の先頭へ	0	^QS	^A
行の末尾へ	$	^QD	^E
次の行へ	j	^X	^N
次のページへ	^F	^C	^V
前の行へ	k	^W	^P
前のページへ	^B	^R	^Z
指定行へ	行G	M G行	
先頭へ	lG	^QR	M <
末尾へ	G	^QC	M >
削除			
カーソル上の文字	x	^G	^D
カーソル前の文字	X	^H or BS	Del
カーソル上の単語		^T	
前の単語		M D	
行末まで		^Y	^K
ファイル関連			
保存	:w	^KS	^X^S
保存して終了	:wq	^KD or ^K^X	
破棄して終了	:q	^KQ	^X^C
編集中すべて保存	:w		^X^S
カーソル位置挿入	:r or : r	^KR	^X^I

・「^」は[Ctrl]キーを押しながら次の文字を，「M」は[Meta(ESC)]キーを押してから次の文字を打つことを示す．
・viは編集モードでのキー入力．このとき，大文字・小文字を区別する．
・Emacsはデフォルトのキーアサイン．
＊vi，WordMaster，Emacsの操作の一部（それぞれが同じ役割とは限らない）

タ用に作られたもっともポピュラーなDOS）用にWordMasterというフルスクリーンエディタがリリースされ，指をくわえて見ていたものである．この間，ラインエディタの使用を強いられていた．それからしばらくして，UX-300Fという東芝のUNIXワークステーションを使ったときは，逆にviの操作体系にものすごくとまどったものである（誰でもはじめてviを使うときはとまどいがあることだろう）．ともあれ，けっこうラインエディタでとおってきた期間が長かったので，はじめてMifesを使ったときのカルチャーショックは大きいものであった．

操作が異なるスクリーンエディタ

フリーソフトウェアとして提供されているポピュラーなテキストエディタには，viのクローンやEmacsのサブセットがある．viもEmacsも，UNIXではもっともよく使われるテキストエディタで，しかもプログラミングの経験を長くもつプログラマが作ったものである．それゆえ，ソースコードを入力・編集するにあたって使いやすいように操作体系が整備されている（……かどうはユーザーが判断するところなのであるが）．また，最近はMifesやFinalなどの市販の高機能なフルスクリーンエディタの操作体系を意識したものも増えてきた．

いずれにしろ，フルスクリーンエディタにはいろいろな種類があり，その操作方法もさまざまである．その中で，UNIXに縁のなかった方にとっては，viというテキストエディタの操作体系は理解の及ばないものであるかもしれない．

viは，それまでのラインエディタを画面で見やすく使えるようにしたもので，ラインエディタのコマンド体系を引きずっているためかモード指向の操作体系になっている．テキストの編集と入力を切り換えて使うのである（viとラインエディタexとの間で行き来もできるようになっている）．また，ファイルの操作はそれまでのラインエディタと同様に行う．通常は編集モードで，カーソル移動，文字の削除，文字列検索などをこのモードで行う．もし，テキストを入力したいのなら，カーソルのある後ろか，直前か，次行の行頭か，前の行の行頭かを考えて入力コマンドとしての文字を入れる．そうしてはじめてテキストが入力できるようになる．画面上の環境を変えるときなどは，ラインエディタのコマンド入力の方法そのものである．

Emacsはviのモード指向に対して，1つのキーを押すごとに機能を呼び出すファンクション実行型のテキストエディタである．たとえば，[A]のキーだけ押せば「aという文字を入力する」というファンクションが起動するし，[Ctrl]キーを押しながら[n]のキーを押せば「画面上の次の行にカーソルを移動する」というファンクションが起動する．見方を変えれば，通常は常に文字列が入力できる状態であるということになる（このほうが自然に感じるのかもしれない）．もし，カーソル移動や文字列の削除，検索などの編集やファイル操作を行いたいのであれば，[Ctrl]キーあるいは[Meta]キー（たいていは[ESC]キーで代用）と併用した文字キーの入力を行えばよい．文字入力以外の操作は，すべてこのような操作で行う．それぞれのキー入力にさまざまなファンクションが対応しており，これらは自由に設定・変更ができるようになっている．

フリーソフトウェアとして提供されているものではMicroEmacsと呼ばれるものが，Emacsに準じたテキストエディタの中ではもっとも長い間使用

FIG 1　図1　[Ctrl]キーと併用するダイヤモンドカーソルの機能例（WordStar）

されてきている．その間に日本語対応などの改良が加えられ，ご丁寧に単一機種専用になった(たとえば矢印キーによるカーソル移動が可能になったとか，VRAMじか書きの超高速スクロールを実現させてしまったとか)ものもある．

Emacsに準じたテキストエディタのいくつかには，キー操作を練習するためのチュートリアルファイルが附属してくる．Emacsのキー操作に慣れない方も，このチュートリアルファイルを使えばすぐに慣れることであろう．

最近リリースされているスクリーンエディタでは，ファイル操作や表示環境の変更がジャンルごとファンクションキーにまとめられ，初心者ににも使いやすいように工夫が凝らされてきている．この類は，作者によってその操作体系がさまざまである．ほとんどのものが単一機種用に作られており，とくに98ユーザーは，選ぶ贅沢を存分に味わうことができるだろう．興味深いのは，これらのスクリーンエディタのカーソル移動がほとんどダイヤモンドカーソルキーに対応していることである．このダイヤモンドカーソルキーは，WordStarやWordMasterから引き継がれているキー操作で，歴史も長い．Mifesや一太郎をはじめとする，市販のテキストエディタやワープロソフトも，この操作方法をサポートしているものも多い．

テキストの1行は？

テキストファイルの「1行」には，2通りの考え方がある．1つは，画面表示やプリントアウトされたときの，見かけ上の行数．そしてもう1つは，記憶装置に記録するときの論理上の行数である．

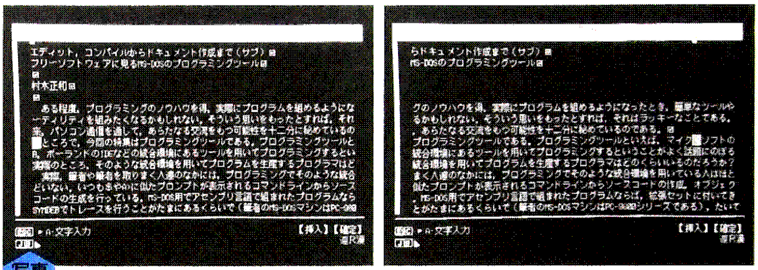


写真1　一太郎：画面全体がそのまま左右スクロールする

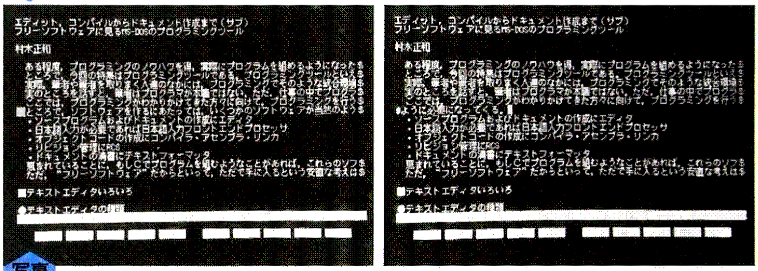


写真2　MicroEmacs：画面の端より先にテキストが続くときは$で示される

見かけ上の行数は，文字どおり画面上で見える行数であり，同じテキストファイルでも，1行80カラムの画面をもつ機種と1行132カラムの画面をもつ機種との間では，全体の行数が違ってくる．これに対して論理上の行数とは，その機種がもつ画面上のカラム数に関係することなく，改行文字単位(MS-DOSなら0D 0Aの2バイト，UNIXなら0Aの1バイト）で考えるものである．たとえば，テキストファイルの先頭から最初の改行文字までは，何文字であれ1行だし，テキストファイルに改行文字を1つも含んでいなければ，これは1行しかないテキストファイルである．

テキストエディタを起動，テキストファイルの読み込みやテキストの入力を行ったとき，もし，論理上の1行が画面上で表示できる1行の範囲を越えてしまった場合，スクリーンエディタは次のいずれかの動作を行う．

①越えてしまった部分を次の行の先頭に送る(折り返す)
②左右スクロールでカバーする
③その両方

①は，ラインエディタやviばかりでなく，市販のMifesやVzなどにも採用されている(Vzなどは折り返すカラム数を指定できるので，どちらかといえば③のケースに近い)．

また，画面上で折り返すスクリーンエディタの場合も，カーソルの移動結果は異なっている．カーソルを上下移動する場合，たいていのスクリーンエディタは画面上の行間で上下移動をする．しかし，viに限っては必ず論理上の行単位でカーソルの上下移動をする．つまり，80カラムの画面で論理上の1行が250文字だった場合，画面上では3行にわたってその内容が表示される．このとき，その行頭にカーソルがあって，[j]キーでカーソルを下に移動した場合，見かけ上は2行分カーソルが移動する．画面上で折り返されても，カーソル移動は1つの行内で移動するように操作しなければならない．

②の場合は，2とおりのパターンがある．一太郎に代表される日本語ワープロソフトでは，左右スクロールする

ときは画面全体がスクロールする．画面が左右スクロールする場合は，誰もがこのようなスクロールをイメージするであろう．これにくらべてEmacsのスクロールはユニークで，カーソルのある行のみが左右スクロールする．画面全体が，メッセージの書かれた紙テープを並べているようなイメージになる．

termcapと環境変数

かつては，フリーソフトウェアはソースコードで供給され，入手したユーザーが各自で自分の機種用にアップデートするようになっていた．現在は各機種用にそれぞれアップデート済みのものを何種類かに分けて提供することが多い．また，ハードウェアに依存したプログラムも増えてきている．

vi, Emacsは汎用性のあるテキストエディタで，できるだけ多くの機種で動くように作られてきた．機種によって異なる表示画面の行数・カラム数・画面制御のコードなどは，termcapというファイルを用いて違いを吸収している．UNIXではホストマシンがtermcapを用意して異機種の端末に対応できるようにしているが，スタンドアロンで使う場合はそれぞれの機種上にtermcapをおいて同じソフトウェアを使うことになる．termcapの要不要に関しては，ドキュメントに必ず示されている．ソフトウェアによっては，このtermcapのサンプルが附属してくるものもある．

MS-DOS上で，termcapが必要なテキストエディタを使う場合，環境変数の設定が必要になる．設定する環境変数はTERMCAPとTERMの2つ．TERMCAPにはtermcapファイルをフルパスで（たいていは¥ETCディレクトリにtermcapを入れ，set TERMCAP =¥ETC¥TERMCAPというように設定する），TERMには使用機種（たとえばset TERM = pc9801というように）を設定する．

setコマンドによる環境変数の設定は，AUTOEXEC.BATに書き込んでおく．AUTOEXEC.BATファイルでの環境設定の例をリスト1に示す．

フリーソフトウェアのテキストエディタ

stevie

viクローンのフリーソフトウェアでは，stevieとelvisがその代表的なものである．とくにstevieはviクローンのフリーソフトウェアの中で，もっともよく知られているものであろう．原版はTony Andrews氏が制作し，パブリックドメインのviクローンとして，ソースコードが提供された．

その後，日本語に対応するように改良されたのはいうまでもないが，日本語対応化を行った人は何人もおり，何種類かの日本語対応版stevie（あるいはjstevie）がある．

よく流通しているものでは，斎藤靖氏が日本語対応化したstevie（vi.exe）や太田純氏が中心となって日本語に対応させたもの（日本語版1.2）などがある．

後者のjstevieにはMS-DOS汎用版（stevie.exe），PC-9800専用版（stevie98.exe，stevie50.exe）がある．stevie98は高速版で，しかもマウスによるカット&ペーストができるようになっている．また，stevie50は98のグラフィック画面を使って50行表示を実現するものである．50行表示は，フォントを圧縮するため，日本語は見づらくなるが，アセンブリ言語やC言語のソースコードを扱うのには便利である．

■ stevie
圧縮ファイル名：jstl3exe.lzh，jstl3p98.lzh，jstl3src.lzh／(C)Tony Andrews／日本語化：太田純(junet: ohta

LIST リスト1　AUTOEXEC.BATの例 1

```
echo off
prompt $P$G
verify on
set LIB=A:¥LIB
set PATH=A:¥WINDOWS;A:¥;A:¥BIN;A:¥LSIC86¥BIN;A:¥DOS
set TMP=A:¥TMP
set TEMP=A:¥TMP
set TERMCAP=A:¥ETC¥TERMCAP
set TERM=pc9801
set TZ=JST-9
set USER=hatchobori
cd ¥USR
```

@src.ricoh.co.jp）ほか

elvis

elvisもまたstevieと並んでよく知られているviクローンの1つである．Steve Kirkendall氏がUNIX用のものを作り，Guntram Blohm氏とMartin Patzel氏がIBM PC用に移植したものを，伊藤純一郎氏がMS-DOS汎用に手直しすると同時に，日本語に対応させた．検索などに正規表現が使えるほか，vi，exのコマンドのほとんどがサポートされている．

elvisがほかのviクローンと異なっているところは，論理上の1行に対する画面上での対応である．オリジナルのviも含めて，この類のテキストエディタは通常，画面上の1行で表示しきれないときは次の行に折り返す．ところがelvisの場合，表示しきれない部分を折り返すことなく，画面からはみ出ているように見せている．画面上で見えない箇所へカーソル移動した場合は，画面全体が横スクロールする．

elvisでは，メインメモリのバッファを越えるファイルも編集できる（バッファに読み込んだファイルのほとんどの部分を一時的にディスクに書き込む）．もし，なにかの事故によって（たとえばATOKで日本語入力中にハングアップしたなど）本体をリセットせざるを得なくなった場合でも，編集中のファイルはディスクに残る．システムが再起動したところで，附属しているvirecを使うことによって復元できる．

■ elvis
圧縮ファイル名：jevl410.lzh，jevl410s.lzh，evl4jd10.lzh／(C) Steve Kirkendall／日本語化：伊藤純一郎（ASCII NET PCS pcs31031）

MicroEmacs

MicroEmacsは，その名前が示すとおりEmacsの縮小版である．本家Emacs（GNU Emacs）の機能は凄い．メイルを送ったりディレクトリの編集をしたり，ゲーム（ハノイの塔など）までできるようになっているのである．Emacsの機能に興味があるのなら，共立出版の「GNU Emacsマニュアル」(Richard Stallman著／竹内・天海監訳）を読まれるといい．

筆者が知るところの最新バージョンは3.9e（J1.3）である．オリジナルはDaniel M. Rawrenceが著作権を所有し，日本語対応は長原宏治氏によって行われている．MS-DOS汎用版，PC-9800シリーズ版，J-3100版がある．各機種専用のものは，ファンクションキーにもそれぞれの機能が割り当てられている．また，PC-9800シリーズ版は高解像度モードもサポートしている．

■ MicroEmacs
圧縮ファイル名：meexe98.lzh，meexe31.lzh，meexedos.lzh，medocs.lzh，mesrcs.lzh／(C) Daniel M. Lawrence／日本語化：長原宏治（ASCII NET PCS pcs12042）

Ng（Micro Nemacs）

Ngは，Mg(Micro GNU Emacs) 2aをもとに，吉田茂樹氏らが制作した日本語対応のEmacsサブセットである．最新バージョンは1.3で，Nemacs（日本語GNU Emacs）3.3.2に準拠している．

ファイル入出力・キーボード入力・画面出力は，それぞれ独立して日本語文字コード体系を扱うことができる．サポートされているコード体形は，JIS，シフトJIS(MS漢字)，EUC(拡張UNIXコード）の3つである．

供給されるファイルは，MS-DOS版，Human68K版，またソースコードを含んだものがある．ソースコードはそのままBSD，あるいはSystem VのUNIXでコンパイルして使えるようになっている．筆者も，このソースコードを入手して仕事場のSystem VにNgをインストールしている．

Cプログラマのために，本家Emacsに備えられているCモードのサブセットが利用できる（Micro Emacsにもある)．このモードにすると，インデントの自動調整や'}'，';'を入力したときに自動改行させたりすることができる．

Dired（ディレクトリエディタ）も装備し，Emacsのサブセットの中ではもっとも高機能なものだといえよう．

■ Ng

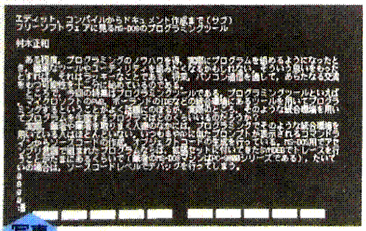


写真3 stevie

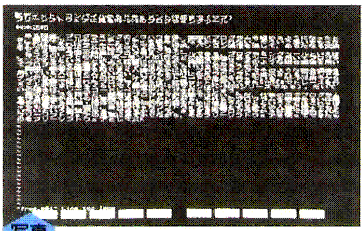


写真4 stevie50

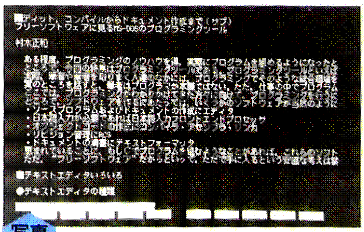


写真5 elvis

おもなUNIXコマンド
dc 逆ポーランド法無限精度電卓

圧縮ファイル名：ngl3doc.lzh, ngl3ex98.lzh, ngl3exe.lzh, ngl3src.lzh／(C)Bob Larsonほか／日本語化：吉田茂樹（shige@csk.CO.JP）ほか

Nit Emacs

Nit Emacs（似てmacs）もまたEmacsを元にして作成されたテキストエディタであるが，この類の中ではもっとも異色のEmacsライクなフルスクリーンエディタといえる．PC-9800シリーズ専用である．

操作方法はEmacsにコンパチブルであるが，矢印キーのカーソル移動の対応はもちろんのこと，[Shift]キーを押しながら上下のいずれかの矢印を押すと，バッファの内容が高速スクロールする．画面は左右スクロールではなく，論理上の1行が画面のカラム数を越えた場合は折り返すようになっており，さらに行末には改行コードが表示されるようになっている．カーソル移動も画面上の位置単位で動く．

完全にPC-9800シリーズ専用に作られているので，termcapファイルは考えなくてもよい．MifesやFinalなどのフルスクリーンエディタを使っていた人にとっても，違和感を感じることはほとんどないだろう．

これは千葉滋氏が独自に制作したもので，最新版は4.1.3である．ただ，筆者は日本語入力フロントエンドプロセッサの都合上，その前の4.1 R2を使っている．

■ Nit Emacs

圧縮ファイル名：nit413.lzh／(C)千葉滋（chiba@is.s.u-tokyo.ac.jp）

SE3（Screen Editor 3）

SE3は，京都コンピュータ学院情報科学研究所が開発した教育用フルスクリーンテキストエディタである．教育現場での使用を考慮して開発されているため，操作体系も画面も数々の工夫が凝らされている．

操作体系はWordMasterやMifesなどを意識し，市販のスクリーンエディタへの移行にも問題はない．動作も非常に高速である．ビジュアルな点でも完成度が高いフルスクリーンエディタといえる．コンパクトで，フロッピーベースでも十分に使うことができる．最新バージョンは3.21Rで，PC-9800シリーズやJ-3100SSなどの機種用のものがリリースされている．ほかにもAXやFMRをはじめとする多くの機種に移植されていくようだ．

論理上の1行が画面よりあふれる場合は横スクロールによってカバーするが，それは256文字までである．それを越えた場合は擬似的な改行を入れ，折り返す．

編集，ファイルなどの操作は[Ctrl]キーとの併用で，WordMasterのそれに準じている．また，ファイル関係は[f·1]キー，[ESC]キーともに作動するようになっている．そのほか，環境設定やブロック編集はファンクションキーを使って操作することができるようになっている．初心者もなじめるテキストエディタといえるだろう．

■SE3

圧縮ファイル名：se32298.exe／(C)京都コンピュータ学院情報科学研究所

ここで紹介したもの以外にも，PC-9800シリーズにはフリーソフトウェアのテキストエディタがいくつか存在する．興味のある方は，アスキーネットPCSのpool msdos, Nifty-ServeのFGAL・アプリケーションやPC-VANのNEC98 byチアリのワードプロセッサ関連のOSLあたりを覗いてみるとよいだろう．

Cプログラマの強い味方

フリーソフトウェアのコンパイラ

日本語に対応しているCコンパイラがフリーソフトウェアとして配布されたときは，一種の驚きと喜びを覚えたものだ．フリーソフトウェアとしてプログラムを開発するのに格好な環境が提供されたのである．それがLSI C-86試食版であった．メーカーが販売するソフトウェアのサンプルを提供するメリットは，ユーザーにとってはそのソフトウェアを試し，活用することができることであり，提供者であるメーカーにとっては，ユーザーの反応をすぐに知ることができる点にある．メーカー，ユーザーお互いの発展のために有効な方法と思われる．

gccとLSI C試食版

フリーで配布されているCコンパイラはいくつかあるようだが，この中でもとくに有名なのはGNU Cコンパイラ（gcc/gpp）とLSI C-86試食版の2つであろう．

gcc（GNU C compiler）は高度なフリーソフトウェアのCコンパイラである．IBM PC用に移植されて使われていたが，最近PC-9800シリーズ用に移植された．UNIXでのプログラミングをしている人にとっては，gccのほうがなじみやすいかもしれない．ただ，

gccはパソコンにとっては大きいものである．CPUが32ビットでなく，しかもRAMを多く搭載していないマシンでは使いにくいだろう．プログラム自体もかなり大きい．

一方，LSI C-86試食版のほうは，エル・エス・アイジャパン(株)が開発・販売している製品「LSI C-86」に制限を付けたものである．制限が付いたといっても，製品版との大きな違いは，スモールモデルしかコンパイルできなくなっているということのみで，簡単なユーティリティやツールを作成するのであれば，これで十分である．試食版のドキュメントにも，「役に立つプログラムの開発をしてほしい」という旨が書かれている．

gccは高度なCコンパイラであるが，入手の容易さや手軽さを考えれば，MS-DOSユーザーにとってはLSI C試食版のほうが取り付きやすいといえよう．なお，LSI C-86試食版の最新バージョンは3.3である．

■ gcc
(C)Free Software Foundation, Inc.

■ LSI C-86 Ver 3.30 試食版
圧縮ファイル名：lsic330.lzh／(C)エル・エス・アイ　ジャパン（株）

リビジョン管理

リビジョン管理の必要性

ソフトウェアを作っているうちに，バグの修正や機能の拡張に迫られたとき，すでにあるソースコードをもとにプログラムを拡張・改良する必要が出てくる．しかしながら，改変したプログラムに新たなるバグが生じたような場合，それは改造どころか改悪という悲惨な結果に見まわれる．このようなときには，改変した箇所をキャンセルして前のリビジョン(版)に戻してやり直すというのが確実な方法である．また，2パターンで機能を比較していきたい場合もあるだろう．その場合，それぞれ版の異なったソースコードを用意しておくのはディスクの無駄使いである．ましてや，1世代前のファイルしか残されないような(＊.BAKのようなバックアップファイルを残して終了するエディタ）環境は役には立たない．そこで，最初に作られたファイルに，版ごとの変更履歴を残すような形で残しておく合理的な方法がある．それを支援するツールがRCS(リビジョン・コントロール・システム)である．

実は，このような支援ツールには，RCSとSCCS（ソース・コード・コントロール・システム）がある．どちらももともとはUNIXのツールであるが，RCSはBSD版，SCCSはSystem III，System Vのものであった．現在のMS-DOSによる開発シーンでは，RCSのほうがよく使われているようである．

フリーソフトウェアのRCS

RCSといっても，RCSという名前のプログラムはない．RCSは，いくつかのツールがひとまとまりになってその機能を実現しているのである．その中にはいくつものツールがあるが，たいていの場合はciとcoだけで何とかなる．

ここまでいってしまうのは，ちょっと野蛮な発言ではあるが，多くのリビジョンを枝分かれさせておく必要がなく，リビジョン履歴を残すだけであれば，この2つだけでことたりる．しかしながら，RCSの機能を本格的に使うのであれば，逆にこの2つだけでは目的が果たせないということもいえる．

MS-DOS用のRCSとしてポピュラーなものは，GNUishのMS-DOS移植版RCSと，T.Tanaka氏の作ったものである．

T.Tanaka氏のものは，RCSファイルに独自の形式を用いている．ドキュメントの説明がしっかりしており，はじめてRCSを使う人にとっても，RCSについて理解しやすいように書かれている．なお，このRCSはシェアウェアである．

GNUishのRCSには，koya氏によって日本語に訳されたマニュアルが附属していた．これにもRCSの各コマンド，RCS入門についての説明がなされている．

■ RCS
圧縮ファイル名：rcs.lzh／(C)T.Tanaka

■ GNU RCS
圧縮ファイル名：gnurcs.lzh／(C) Walter Tichy／MS-DOSへ移植：koya

(koya@math.titech.ac.jp)

ドキュメント作成にあたって

日本語の入力

プログラムが(ある程度)完成し，それをフリーソフトウェアとして提供するのなら，どうしてもドキュメント(マニュアル)を作成する必要が出てくる．たいていの場合，ドキュメントは日本語で書くことになる．とくにPC-9800シリーズやJ-3100シリーズ，FM R シリーズなど，国産機種で動くように作られたものでは，日本語で書いたほうがよいに決まっている．

たいていのプログラマは，VJEやATOKをはじめとする市販の日本語入力フロントエンドプロセッサを使っていることであろう(よけいなお世話だが，ATOKは一太郎以外で使うときはハングアップのリスクを背負うことになる)．

フリーソフトウェアとして提供されている日本語フロントエンドプロセッサとしてWXPがあることはすでに述べたが，WXPはもともとドキュメントプロセッサ「創文」に付属していた変換効率に定評のあったWXや，統合ソフト「カドリール」に付属しているWXSをフリーソフトウェア用に作り直したものである．フリーソフトウェアとして流通させたため，ユーザーの数多い意見を取り入れることができ，かなりよいものに変わっていった．それが大きく発展して，現在は「WX II⁺」と，これをMS-Windows 3.0専用にした「WX II Win」というものが市販されている．

WXPは，フリーソフトウェアでありながら実用に耐えるすばらしい日本語入力フロントプロセッサである．余談だが，このWXPは最近ブラザーのソフトベンダー「武尊（TAKERU)」において1,500円で販売されていた．

■日本語フロントプロセッサ WXP
圧縮ファイル名：wxp.lzh，wxpdic.lzh／(C)エー・アイ・ソフト（株）

読めない文字に注意

文字はコードでディスクに記憶されたり，通信回線に送り出されたりする．ところが，文字コードは各機種共通のものと，そうでないものがある．とくにPC-9800シリーズは共通でないコードに文字が定義されている．丸つき数字やギリシャ数字，罫線キャラクタなどがそうだ．これらの文字をドキュメントに使ってしまうと，PC-9800シリーズおよびエプソンの互換機以外で読むことができなくなる．

ドキュメントはどの機種でも読めるものである（たとえソフトウェアがPC-9800シリーズ専用であろうとも！)．したがって，これらの文字を使うことは極力避けたほうがよい．しかし，慣れないユーザーにとっては，どの文字が共通でないかを判別するのは容易なことではないだろう．

serow氏のraskは，これらの文字を知らせてくれる便利なツールである．ドキュメントを作成する場合にこのraskを起動しておけば，MS-DOSのtypeコマンドやcatなどで表示したとき，色を変えたりして(属性は自由に設定できる)表示してくれるのである．raskはもともと第2水準漢字がそれほど普及していない時代に作られたもので，第2水準漢字を知らせるのがおもな目的だったが，PC-9800シリーズ独自の文字も知らせてくれるので，現在でも利用者は多い．

また，筆者のdenecはPC-9800シリーズ独自の文字をどの機種でも読める文字の集まりに変換するフィルタである．詳しいことは，昨年10月号の「オンラインソフト採集帳」に示されている．

■ rask
圧縮ファイル名：rask.lzh／(C) serow（ASCII NET PCS pcs19614)

■ denec
圧縮ファイル名：denec.lzh／(C)八丁堀（ASCII NET PCS pcs18303)

テキストフォーマッタの利点

プログラマなら，たいていの場合，ドキュメントはテキストエディタを使って入力する．一般に日本語ワープロソフトと呼ばれているものを使ってドキュメントを作成してもかまわないのだが，ワープロは紙に印刷して見るというものである．フリーソフトウェアの場合，ドキュメントファイルは画面上に表示して見ることが多い．

テキストフォーマッタは，テキストエディタなどで作成された文書ファイルを整形するものである．文書入力時は最終的な整形結果をイメージしながら文書を入力する(画面でレイアウトは確認しない)．したがって，ワープロに慣れているユーザーにとって，テキストフォーマッタを通すという考え方は，理解の及ばないところがあるかもしれない．

テキストフォーマッタの利点は，追い込みが手軽にできるということであろう．たとえば，図2の文書を図3に

変えたいときのことを想定してみよう．

この作業を，ワープロやテキストエディタだけで行うのは非常にめんどうである．インデントのスペースを入れたり消したりしなければならない．もしかすると，最終版のドキュメントが完成したとき，ずれてしまっているかもしれない．

テキストフォーマッタは，段落やインデント，１行の文字数を自動的に調整してくれる．このため，テキストエディタでプログラムの入力作業を行うときは，文字数変更の細かい調整（空白の削除や挿入など）を気にかけることなく，入力（ライティング）作業に集中することができる．テキストフォーマッタにかける文書は，埋め込みコマンドというものを行単位で入力することによって，テキストフォーマッタによる整形を実現する．

テキストフォーマッタを通すテキストをテキストエディタで入力する場合，改行はどこに入れてもかまわない．最終的にヘッダ，フッタ，マージンを調整して出力してくれるからである．どんなに変なところに改行が入っていても，段落の頭の文字（たとえば改行と空白）が入っていれば，そこで改行してくれる．あとは，自動的に不要な改行部分を詰めてくれる（もちろん，強制改行させることも可能）．

ワープロソフトは単一機種対応のものが多いし，単にテキストファイルとして文書を作成し保存するには無駄な機能が多かったり，遅かったりする．昔，一太郎の文書ファイルをそのままドキュメントファイルにしてしまい，ほかのMS-DOSマシンでドキュメントが読めなかったということがあった（筆者はフィルタを作って回避したが）．

とくに，多機能なワープロソフトほど遅いという傾向がある．テキストエディタとして使うのには重すぎるであろう（VJE-Penなどは，ワープロでありながらこのあたりのことをよく考えて設計されていると思う）．

フリーソフトウェアのテキストフォーマッタ

fin/efin

UNIXのワードプロセッシングシステムは，テキストエディタで文書を作成し，roff，nroff，troff，TeXなどのツールで整形してこれを出力するというものである．nroffはテキストレベルでの整形と，文字単位での装飾機能をもっているので，たいていのパーソナルコンピュータで同じ機能を実現することができる．

nroffは英文ワードプロセッシング用に作られていたため，基本的には日本語に対応していなかったが，アスキーネットが実験段階だったころ，花田氏が日本語対応のテキストフォーマッタjroffを提供した．この仕様に，山田祥平・日下部陽一両氏が拡張を加えたものがfinである．

finは英文に関する機能が弱く，ワードラップやライトジャスティフィケーション（右端そろえ）を行ってくれなかった．そこで，公開されたfinのソースコードにfuzzy_No.3氏が手を加えて強化したのがefin（Expanded fin）である．当然，efinはfinとの互換性が保たれている．

fin/efinには，拡張用のコマンドやFAX，メイル用の定型マクロが用意されており，これを指定のディレクトリに入れて使うことができる．マクロファイルは通常，ドライブAの¥USR¥LIB¥FMACディレクトリにあるとみなされるが，このディレクトリパスを変えたければ，環境変数FMACに指定のディレクトリをフルパスで設定する．

■ fin

圧縮ファイル名：fin118.lzh，finsrc.lzh／(C)syohei（ASCII NET PCS pcs00500）ほか

■ efin

圧縮ファイル名：efin110e.lzh，efin110s.lzh／(C)fuzzy_No.3（ASCII NET PCS pcs21322）ほか

FIG 図2 2

b　ファイルをバイナリで
　　オープンするオプショ
　　ンです．

FIG 図3 3

b　ファイルをバイナリ形
　　式でオープンするオプ
　　ションです．

FIG 4　図4　テキストフォーマッタを通す前の文書

```
.ll 72
.mh 'ZHECONV(1)'Pre-Release'ZHECONV(1)'
.mf 'Page %''(Updated 09/23/91)'
.in 5
.ti 0
NAME
.br
zheconv - 全角文字・半角文字および EUC・MS 漢字コードの相互変換。
.sp
.ti 0
SYNOPSIS
.br
zheconv [-z|-h] [-e|-m] [-i] [ファイル...]
.sp
.ti 0
DESCRIPTION
.br
このプログラムは jctype.h、jstring.h をサポートしてない機種で、
zentohan、hantozen 関数を利用するためのライブラリを
実行形式のフィルタにしたものです（本当は
関数がちゃんと機能するかどうかのテストしたかっただけだったりして）。
.sp
なお、フォントが固定されていて、
1バイト文字が 2バイト文字の半分の大きさで表示される
時代遅れの端末（ウチのマシンもそうだ、ワッハッハ）に合わせて、
1バイト文字を「半角文字」、
2バイト文字を「全角文字」と便宜上呼ぶことにします。
.br
NEC などは 24ドットシリアルプリンタ出力のために 2バイトの半角文字を
漢字 ROM に持っているようですが、
そんなローカルなものは相手にしません。
.sp
ファイルで対象となる文字は、
数字、英大文字、英小文字、特殊記号（ASCII コードの図表で示される文字）
の半角および全角文字のみです。
zentohan、hantozen 関数は、もともとカナをサポートしないので、
このプログラムでもサポートしません。
.sp
このプログラムはオプションを指定しなければ
いずれの文字の変換も行われません（cat がわりとして使えそうだ）。
.br
文字または文字コードの変換を行いたい場合は、
以下のオプションを指定します。
.sp
.ti 0
OPTIONS
.br
オプションはハイフンに続く 1文字のアルファベットからなります。
.sp
.in +8
.ti -8
-h      全角文字があれば半角文字に変換します。
.br
.ti -8
-z      半角文字があれば全角文字に変換します。
.br
.ti -8
-e      MS漢字（シフトJIS）コードを EUC に変換します。
.br
.ti -8
-m      EUC を MS漢字（シフトJIS）コードに変換します。
.br
.ti -8
-i      2バイト文字に不正コードがあっても実行を中断しません。
.in -8
.sp
.ti 0
BUGS
.br
1バイトカナが -e オプションでも EUC に変換できません。
まぁ、使用にあたって差し支えはないことでしょうが...。
.sp
.ti 0
AUTHOR
.br
はっちょ。
.sp
.ti 0
TECHNICAL INFORMATION
.br
もし、EUC のマシンで使うのなら、
ソースコードそのものを MS漢字から EUC に変換して（MS-DOS マシンが手元にあ
実行形式のファイルで zheconv -e zheconv.c として
変換したものホストに転送することもできるかもしれない）
再びコンパイルをかけてください。
.sp
なお、実行ファイルを作成する場合は、
ヘッダファイル "ohcta.h" を用意し、
`MAINMODULE' を定義してコンパイルしてください。
.sp
.ti 0
COMMENT
.br
.ig
このプログラムは実行形式のファイル "zheconv.exe" を除いて PDS 同様、
すべての権利を放棄しようと思っています。

実行形式のファイルは LSI C-86 試食版
（製品じゃないんだから）
でコンパイルされて
います。
```

FIG 5　図5　テキストフォーマッタに通して印刷した文書

```
ZHECONV(1)                  Pre-Release                  ZHECONV(1)

NAME
     zheconv - 全角文字・半角文字および EUC・MS 漢字コードの相互変換。

SYNOPSIS
     zheconv [-z|-h] [-e|-m] [-i] [ファイル...]

DESCRIPTION
     このプログラムは jctype.h、jstring.h をサポートしてない機種で、zent
     ohan、hantozen 関数を利用するためのライブラリを実行形式のフィルタに
     したものです（本当は関数がちゃんと機能するかどうかのテストしたかった
     だけだったりして）。

     なお、フォントが固定されていて、1バイト文字が 2バイト文字の半分の大
     きさで表示される時代遅れの端末（ウチのマシンもそうだ、ワッハッハ）に
     合わせて、1バイト文字を「半角文字」、2バイト文字を「全角文字」と便宜
     上呼ぶことにします。
     NEC などは 24ドットシリアルプリンタ出力のために 2バイトの半角文字を
     漢字 ROM に持っているようですが、そんなローカルなものは相手にしませ
     ん。

     ファイルで対象となる文字は、数字、英大文字、英小文字、特殊記号（ASC
     II コードの図表で示される文字）の半角および全角文字のみです。zentoh
     an、hantozen 関数は、もともとカナをサポートしないので、このプログラ
     ムでもサポートしません。

     このプログラムはオプションを指定しなければいずれの文字の変換も行われ
     ません（cat がわりとして使えそうだ）。
     文字または文字コードの変換を行いたい場合は、以下のオプションを指定し
     ます。

OPTIONS
     オプションはハイフンに続く 1文字のアルファベットからなります。

     -h      全角文字があれば半角文字に変換します。
     -z      半角文字があれば全角文字に変換します。
     -e      MS漢字（シフトJIS）コードを EUC に変換します。
     -m      EUC を MS漢字（シフトJIS）コードに変換します。
     -i      2バイト文字に不正コードがあっても実行を中断しません。

BUGS
     1バイトカナが -e オプションでも EUC に変換できません。まぁ、使用にあ
     たって差し支えはないことでしょうが...。

AUTHOR
     はっちょ。

TECHNICAL INFORMATION
     もし、EUC のマシンで使うのなら、ソースコードそのものを MS漢字から E
     UC に変換して（MS-DOS マシンが手元にあれば、実行形式のファイルで zh
     econv -e zheconv.c として変換したものホストに転送することもできるか
     もしれない）再びコンパイルをかけてください。

     なお、実行ファイルを作成する場合は、ヘッダファイル "ohcta.h" を用意
     し、`MAINMODULE' を定義してコンパイルしてください。

COMMENT

Page 1                                           (Updated 09/23/91)


ZHECONV(1)                  Pre-Release                  ZHECONV(1)
     実行形式のファイルは LSI C-86 試食版（製品じゃないんだから）でコンパ
     イルされています。また、euctojms、jmstoeuc 関数は、LSI C-80 クロスコ
     ンパイラのライブラリのソースを参考にさせていただきました。

     このプログラムおよびドキュメントの品質、性能および利用者の誤操作に関
     するリスクはすべて利用者が負うものとし、作者には何の責任・拘束もあり
     ません。また、これらによって利用者に損害が生じたとしても、作者は一切
     の責任を負いません。
     改変したときのリスクは、その改変者がリスクを負うことにもなります（当
     然の助動詞）。
     このソフトウェアのサポートや質問は気まぐれで応じます。
     なお、このプログラムとドキュメントはテストバージョンです。

POINTLESS RAMBLING
     眠い！


Page 2                                           (Updated 09/23/91)
```

ntf

ntfはjroffやfinを意識し，作表などの機能を取り入れてPoorTom氏が完成させたテキストフォーマッタである．コンパクトな本体と強力なマクロ機能によって，比較的小さなプログラムで本格的なテキストフォーマッタが実現されている．

本体は英文・欧文混在の文書を整形するものとして使用できるが，マクロを使うことによってntfがもつ強力な機能を発揮できる．章立て，箇条書，表組，脚注，参考文献 などの機能は，マクロによって実現されており，そのまま活用できるものが用意されている．ntfで評判になっている機能のほとんどは，このマクロレベルで実現されているもので，ユーザーが比較的簡単に追加・変更できる．

対応プリンタも豊富で，標準でNEC，IBM，ESC/Pなどに対応している．

ソースコード，マクロ，ドキュメントはすべて公開されており，とくにプログラムは日本語への対応がフレキシブルにできるように組まれているため，移植がしやすい．このため，改造したり新たなプリンタに対応させたりして活用しているntfユーザーもいるようだ．

■ **ntf**

圧縮ファイル名：ntf20.lzh／(C)Poor Tom（ASCII NET PCS pcs14235）

XTR

XTR（teXT　processoR）は，nroff，troff，finなどとはまた違う特徴をもったテキストフォーマッタである．もともとは横幅の調整をしてテキストの形を変えたり，文字列を置き換えて出力するフィルタとして設計されたteXt TRansformerを現在のテキストフォーマッタとして発展させたもので，その整形のための機能は豊富である．もちろん，従来のテキストフォーマッタのように埋め込みコマンドを組み込んで整形する機能ももっているが，nroffやfinのそれとは異なっている．

いちいちインデントを指定しなくても，自動的にインデントするオートインデント機能を備えているのは大きな特長であろう．

入力するテキストの形式は問わず，整形済みテキストファイルでもかまわない．はじめから整形されているものであっても整形しなおしたり，元の改行を入れない形式に戻したりすることもできるのである．和文・欧文両用であり，ワードラップやジャスティフィケーションの機能も充実している．

XTRは単なるテキストフォーマットではなく，バイナリファイルにも対応する文字列の置き換え機能や1画面ごとに停止して表示するといった，画面表示のためのツールとしても有効な使い方ができる．何も指定しないのなら，ただのtypeコマンドと同様に動作する．これを，1行の文字数を指定したり，テキスト内の埋め込みコマンドを駆使することによって，ワープロを使わずにきれいな文書を作成することが可能になる．

■ **XTR**

圧縮ファイル名：xtr113a.lzh／(C)村上真雄（NIFTY-Serve QGA03215）

tbl

tblは，これまで紹介したテキストフォーマッタとは用途が異なっている．tblは作表のためのツールで，特定の文字で区切られた項目をもとに作表するものである．ntfにはこの機能があるが，多くのテキストフォーマッタには作表の機能がない．

tblももともとUNIXで作表を行うためのツールで，nroffとともに使われている．MS-DOSへはakiii氏が移植し，罫線として使うキャラクタが何種類か選べるようになっている．最新バージョンは1.28でソースコードも公開されている．

tblの便利なところは，それぞれの項目の文字数を意識しなくても，自動的に作表してくれることである．ワープロなどで作表を行うときは，それぞれの項目の文字数を数えて罫線を引くか，あるいは罫線のところを空白を開けて文字を入れるといっためんどうな作業をしなければならない．ところが，tblはこのめんどうな作業を省略することができる．もちろん，各項目間の文字数を指定したりすることも可能である．

tblは，あらかじめtbl用に作成したテキストファイルを変換して表組みの形で出力するが，finまたはxtrの埋め込みコマンドを含めたテキストファイルを出力させることもできる．

■ **tbl**

圧縮ファイル名：tbl128.lzh，tbs128.lzh／(C)岩本明（ASCII　NET　PCS pcs31549）

最後に

以上，簡単ではあるが，ソフトウェアを作成・公開するまでに必要なMS-

DOS用のフリーソフトウェアをいくつか紹介してきた.

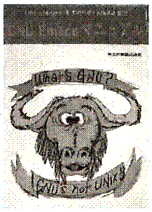
筆者は，これらのツールをアスキーネットPCSから入手している.筆者がフリーソフトウェアを作成するにあたっては，まずelvisでソースコードを書き，LSI C-86試食版でコンパイルして，エラーがなかったらRCSのciでリビジョン登録してから，再びコンパイルする.

ドキュメントはelvisかNit Emacsで入力し，tblとfinまたはefinで整形している（最近ntfも使いだした).筆者がテキストフォーマッタにfin/efinやntfを使うのは，埋め込みコマンドが，かつて使っていたnroffと同じだからである．今までテキストフォーマッタに慣れ親しんでいない方には，xtrも興味深いだろう.

もし，読者の方々が最近プログラミングをマスターし，ちょっとしたフリーソフトウェアを提供してみたいと思うようになったとき，また，プログラムを組んだことがなくて，Cのプログラミングの練習をしてみたいと思っている，そういった方々にとって本記事が参考になれば幸いである.

書籍案内——フリーソフトウェアを知るために

■GNU Eamcs マニュアル



R.Stallman 著，
竹内郁雄天海良治
監訳
2,781円
共立出版

GNU Emacsは，R.StallmanとFSF(Free Software Foundation，Inc.)によって配布が行われている高機能なエディタであり，この本はそのマニュアルの翻訳である．GNUは，UNIX互換のシステムを誰もが自由に入手することを可能にしようとする壮大なプロジェクトである．コピーの自由を保証したcopyleftの概念や配布方針を示した一般公有使用許諾書は一読に価する.

■インサイド GNU C コンパイラ

フリー・ソフトウェアファンデーション編著
岩谷宏，都田克郎
共訳
2,300円
啓学出版

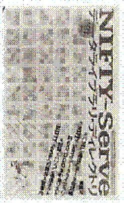
この本は，GNU CC Ver.1.26のマニュアルを翻訳・製本したものである．したがって，GNUのcopyleftに従い，出典を明記することで複写・再配布が可能となっている．GCCは機能の高いCコンパイラであるが，慣れない人間にとっては使いにくい面がある．そのマニュアルが日本語で読めるようになったことの意義は大きいといえるだろう.

■PC-9801版 MS-DOS 定番フリーソフトウェア集

月刊アスキー編集部編
2,980円
アスキー

LHA，FD，history，xscriptをはじめ，定番とも呼べる代表的なフリーソフトウェア24本を紹介した本．3.5インチ2HDと5インチ2HDのフロッピーディスクが付属し，それぞれに全掲載ソフトウェアが収録されている.

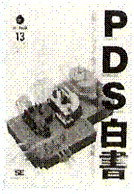
■Nifty-Serve データライブラリ・ディレクトリ('91年版)



2,000円
発行：青峰社／発売：星雲社

パソコン通信ネットワークNIFTY-Serveのすべてのフォーラムについて，データライブラリのインデックスを集めた本．インデックスは'91年8月12日～14日の時点のもので，番号／バイト数／属性／データ名が記載されている.

■PDS 白書



SE 編集部編
1,500円
翔泳社

PDSという形ではじまった公開ソフトウェアをめぐって11人の論者がその歴史的背景，現状と今後の方向性などについてさまざまな角度から考察する．合わせて40本の公開ソフトウェアによるモデルシステムが紹介されている.

■フリーソフトウェア派宣言

FGAL 編
1,500円
翔泳社

NIFTY-Serveのソフトウェアギャラリー(FGAL)を運営するメンバー自身によるフリーソフトウェアガイドブック．ソフトの探し方，ダウンロードの実際などのハウツーから，90本のソフトウェアレビュー，ライブラリリストなど，FGALのすべてを紹介する.

ソフトウェア・ツールや開発ツールという言葉は，道具のように役に立つプログラムを意味する言葉として，現在ではコンピュータに関わる人のあいだでごく普通に使われている．しかし，何にでもはじめはある．ソフトウェア・ツールという言葉を世に広めたのは，『Software Tools』という本であり，その本では，後にUnixのcpやwc，grep，edなどとなるツール類が紹介されていた．ソフトウェア・ツールという発想は，Unix上で開花することとなり，さらにMS-DOSなどの環境にも引き継がれていく．1冊の本にはじまるソフトウェア・ツールの流れを，『Unix道具考』などの著書で知られる祐安重夫氏にたどっていただいた．

特集
プログラミングツール大集合

プログラムをつくるための道具の系譜
ソフトウェア・ツール，UNIX，MS-DOS

祐安重夫

PROGRAMMING
TOOL

ソフトウェアツールのはじまり

ちょうど15年前のことになるが，丸善の洋書売場で本棚をながめていたとき，奇妙な題名の本が目についた．ためしにとりだしてページをめくってみたところ，おもしろそうな内容だったし，当時のコンピュータの本にしては組版がきれいでよみやすそうだったので，購入した．本の題名は *Software Tools* といって，Addison-Wesleyの新刊だった．今ではあたりまえのものとなってしまったソフトウェア・ツールという言葉は，このときはじめて世の中にあらわれたのだった．

この本のまえがきによると，ソフトウェア・ツールとは「道具として役に立つよいプログラム」を意味しているという．たとえばFortranで数千行のプログラムを別のコンピュータに移植したいとき，format文を抽出してそれが移植先のコンピュータに適合しているかどうか確認する必要があったとする．これはテキストの中から特定のパターンをみつけるという問題に一般化できる．そのような仕事をおこなうプログラムを用意しておけば，それ以外の仕事にも便利に使用できることになる．そのようなプログラムをソフトウェア・ツールとよぶのである．

この本の中のプログラムは，Ratforという名前のプログラミング言語で記述されている．RatforはFortranに構造化構文を追加したもので，プリプロセッサによってFortranのプログラムに変換して処理するようになっていた．この本の中にはRatforからFortranへの変換プログラムも，ソフトウェア・ツールのひとつとして掲載されている．

当時Fortranは広範囲のコンピュータで使用できる，もっとも代表的なプログラミング言語の一つだった．そのFortranで処理できることで，この本に掲載されたソフトウェア・ツールはおおくのコンピュータに移植され，実用化されていった．

現在ならこのような本のプログラムは，C言語によって記述されるのが普通だろう．しかし当時C言語の使用できる唯一のオペレーティング・システムであるUnixは，まだ一般にはほとんど知られていなかった．日本で最初にUnixを紹介した論文が発表されるのは，この本が出版された翌年の1977年のことである．

Software Tools の著者はB.W. KernighanとP.J.Plaugerの二人だが，Kernighanはベル研究所でUnixの開発にかかわってきた人物であり，1978年にはC言語の開発者であるD.M.Ritchieとともに，*The C Programming Language* という本をだす．つまり *Software Tools* という本は，Unixの開発の過程でうまれた技術や概念を，より一般的なかたちで提供するという意味があったのだ．

この本のなかのプログラムは名前こそかわっているが，Unixのcpでありwcでありgrepでありedである．それ以外にもUnixの主要なコマンドのおおくが，Ratforで記述されて掲載されている．この本はUnix上で編集されtroffによって写植機に出力されたものを版下として使用しているのだが，本の中にはtroff(というよりタイプライタ出力を前提にしたnroff)のサブセットも掲載されている．

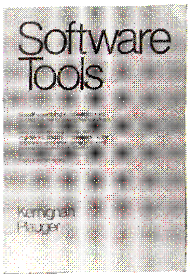


写真1　Software Tools

この時点ではUnixがこれほど普及するとは，誰も予測していなかったにちがいない．しかしこの本のもつインパクトはおおきかったし，またC言語もUnix以外のOSに移植されるようになってきた．アメリカでは大学を中心に，Unixがひろくつかわれるようになってきたこともあって，当時出現した8ビットのパーソナル・コンピュータにこの本に掲載されたようなツールを移植することが，すこしずつおこなわれるようになってきた．いささか本末転倒しているが，Ratforで記述されたものをCで書き直して，パーソナル・コンピュータで動作させるといったこともおこなわれた．

その後時代がうつりかわり，Unixがワークステーションの標準OSとなり，16ビットや32ビットのパーソナル・コンピュータでもUnixが動作するようになった．またMS-DOSにも，Unix上の便利なツールが移植されてきているが，これにはさきほどのべたような背景があったのだ．

ソフトウェアツールの意味

Software Tools は原書が出版されてから5年後に，日本語に訳された．

それが『ソフトウェア作法』(木村泉訳，共立出版)という本である．この『ソフトウェア作法』という邦題は，原書の精神をじつに的確につたえている．この本は便利なソフトウェア・ツールをソース・プログラムでしめすだけではなく，よいプログラムのつくりかたにも重点をおいていたからである．

たとえば構造化プログラミング(structured programming)，下向き設計法(top-down design)，段階的詳細化(stepwise refinement)，構造設計法(structured design)，協調プログラミング(egoless programming)といったソフトウェア工学上の概念を，実際によいプログラムを作成する過程をとおして，具体的かつ実用的なものとしてしめしている．さらに，そのようにしてつくられたソフトウェア・ツールが，よいプログラムをつくるための道具となっているのである．

最初にプログラミングのもっとも基本的な道具である，プログラミング言語を選択しなければならないのだが，この本ではその時点でもっとも広範囲のコンピュータに適合できるという点から，Fortranを選択する．ただFortranは，構造化プログラミングのための道具としては，いささか不満がある．そこでRatfor(Ratinal Fortran――合理的Fortran)という，構文の拡張をおこなう．この言語の処理系をあたらしくつくるのは負担がおおきいので，RatforからFortranへの変換をおこなうソフトウェア・ツールをつくることにする．といった具合に，できるかぎり効率よくバランスをとりながら，道具をととのえていく．

この本ではもう一つ，現在ではあたりまえとなっている概念がしめされている．それはおおくのソフトウェア・ツールが，入力データに変換をおこなって，出力データとするフィルタとして構成できるということである．個々のフィルタは簡単なものであっても，それをくみあわせてつかうことで，非常に複雑な処理をおこなうことができるのである．

ひとつのプログラムに必要な機能をすべてつめこむのではなく，それを汎用性がありコンパクトなツール群に分割することで，より柔軟で多様な状況に対応できるプログラムの道具箱をつくることができる――これがソフトウェア・ツールのひとつの形なのである．

もちろんフィルタとして構成できないような性質の，ソフトウェア・ツールも存在する．それをこの本では，環境と接触するためのプログラムと位置づける．ファイルやプリンタといったものをあつかうプログラムが，そのようなツールである．これを統一的にあつかう方法は，ちょうどC言語の入出力ライブラリに相当するものを用意することで解決される．現在MS-DOSなどで動作するC言語の処理系が，Unix上と同等の入出力ライブラリをもつことで，プログラムの移植性がたもたれているのとおなじである．

この本の最初の部分ではコピー，文字数(単語数，行数)のカウント，タブの処理，文書の圧縮と復元，暗号化，文字の置換，数の変換といった現在ではおなじみのツールと，フィルタを連結するためのパイプラインの概念がしめされる．

またソート，パターン照合といったツールについては，それぞれ1章がさかれている．パターン照合というのは，最初に例としてあげたテキストから特定のパターンを抽出するためのツールである．この本ではfindという名前がつけられているが，いうまでもなくUnixのgrepのことである．ここではgrepの重要な特徴である正規表現と，その実現方法がしめされている．

さらにプログラミングにとって，プログラミング言語とならんで重要な道具であるエディタが，つづいて紹介される．といっても現在主流のスクリーン・エディタではなく，Unixのedに相当する行エディタである．この時代には紙テープやパンチ・カードもまだつかわれており，すべてのコンピュータにエディタのようなツールが用意されていたわけではなかったのだ．またエディタがあったとしても，最低限の機能しかなかったり，コマンド体系が複雑でつかいにくいものもかなりあった．ここでつくられるエディタは，正規表現をつかった検索や置換の機能があり，コマンド体系も一貫してつかいやすいものになっている．さらにあらかじめ用意されたコマンド・ファイルをつかって，フィルタとしてもつかえるようになっている．

次に紹介されるのは，文書整形ツールである．これはUnix上のnroffのサブセットにあたるもので，現在のワードプロセッサのような，画面上で確認しながら文書を作成していくものではなく，テキスト内に整形のためのコマンドをうめこんで，それを処理して出力するという形式のものである．現在でもUnix上では，nroffやtroffあるいはTEXといった文書整形システムが，ひろくつかわれており，画面上で確認していく形式のワードプロセッサの方が，かならずしもつかいやすいわけではない．

つづいての章はマクロ・プロセッサであり，Unixのコマンドでいえばm4にあたる．C言語のプリプロセッサをより一般化したようなものであり，各種のプログラミング言語の機能拡張ツールとして使用できる．これにはRat-

おもなUNIXコマンド
fmt　簡単なフォーマッタ

for導入のための準備といった意味もあり，つづく最後の章では，Ratforから Fortranへの翻訳ツールがしめされ，完結する．

こうして *Software Tools* という本は，Unixのエッセンスを紹介し，ソフトウェア・ツールという言葉を世の中にひろめることに成功したのだが，その後Unix自身が予想外に普及していくことになる．各社のミニコンピュータやメインフレームにUnixが移植され，それにともなって8ビットのマイクロコンピュータをふくむ小規模なシステムにも，Unix風のOSが開発されていく．

16ビットのパーソナル・コンピュータの時代になって，標準OSとしての地位を獲得したMS-DOSは，Unix風の機能拡張をおこなっていったし，16ビットのパーソナル・コンピュータにもUnixが移植されていった．またUnix上の便利なコマンドと同等のものがMS-DOSなどの他のOSのためにつくられ，商品として販売されるだけではなく，フリーソフトウェアとして公開されている．

今では *Software Tools* に掲載されたツールをそのまま移植する必然性はほとんどなくなっているが，そこでしめされたソフトウェア・ツールの思想は今でも有効である．逆にあまりにも簡単に様々なツールが使用できる環境になってしまったために，ソフトウェア・ツールの思想がわすれられつつあるのが，現在の状況かもしれない．

Unixとソフトウェア・ツール

ソフトウェア・ツールという概念をうみだすもとになったUnixにおいて，どのようなツールがあるのかを，ここで紹介しておこう．Unixシステムには通常数百個のコマンドがふくまれている．そのすべてがソフトウェア・ツールというわけではないが，おおくのコマンドはソフトウェア・ツールとしての一貫性をうしなわないようなつくられかたをしている．

表1に主要なコマンドをしめす（一部にフリー・ソフトウェアをふくむ）．

これらのうちの一部に対応するものは，MS-DOSの標準コマンドにもある．またそれ以外でもMS-DOS上で実現され，販売されたり無料で配布されているものがある．ただ残念なことに，MS-DOS上ではUnixほどには一貫したかたちで，これらのコマンドをくみあわせることができない場合がある．

またパイプラインによるフィルタのくみあわせという点では，マルチ・プロセスのOSであるUnixと，基本的には一人のユーザが一度にひとつのコマンドを実行するOSであるMS-DOSでは，実現方法や効率がかわってくる．

たとえばgrepでファイルから特定のパターンを抽出して，それをソートすることをかんがえてみよう．Unix互換のsortとgrepがMS-DOS上でもつかえるなら，どちらのOSでも，図1のようなコマンドを実行することになる．“|”がパイプラインをあらわす記号で，図2のようにすると，コマンドfooの出力（単独で実行すれば画面にでる内容）が，コマンドbarの入力（単独で実行した場合は，キーボードから入力するもの）となる．

“|”の処理はUnixではshやcshなどのシェルが，MS-DOSではcommand.comがおこなうが，実際にはその処理の過程はまったくことなっている．MS-DOSの場合には上記のコマンドの実行は，図3の二つのコマンドを順次実行することと同等である．“>”は出力を指定したファイルにおくることを，“<”は入力を指定したファイルからうけることを意味している．

Unixの場合もshやcshが“>”や“<”を処理できる（というより，MS-DOSのcommand.comがUnixのシェルからこの機能をうけついだ）ので，おなじようにしてもいいのだが，実際にはパイプという外部からみえない特殊なファイルを利用して，並列に処理される．

FIG 図1 1

```
grep  pattern  file  |  sort
```

FIG 図2 2

```
foo  |  bar
```

具体的には pipe というシステム・コールによって，二つのファイル・ディスクリプタをつくる．この二つはそれぞれ入口と出口になっており，入口に出力した内容はそのまま出口から入力できる．このままではパイプにはあまり意味がないようにみえるが，Unix では fork というシステム・コールでプロセスを二つにわけることができるので，そこでパイプの機能が重要な意味をもつ．

fork を実行してもう一つプロセスをつくる．その中でファイル・ディスクリプタの 1 番（標準出力）を close し，パイプの入口を 1 番にきりかえる．その上でコマンド foo を実行する．

もう一回 fork を実行し，さらにプロセスを一つつくる．そこではファイル・ディスクリプタの 0 番（標準入力）を close し，パイプの出口を 0 番にきりかえる．その上でコマンド bar を実行する．sh あるいは csh は二つの子プロセス（foo と bar）の実行が終了するのを待つ．この処理をおこなうプログラミングは，図 4 のようなものとなる（実際のシェルの中では，もうすこし複雑な処理がおこなわれる）．

CHART 表 1

ファイル処理	
cat	複数のファイルの連結
cp	ファイルのコピー
rm	ファイルの削除
mv	ファイルの移動
cmp	バイナリ・ファイルの比較
diff	テキスト・ファイルの比較
diff3	3つのファイルの比較
grep	パターンの照合
more	ページ単位でのファイルの表示
sort	ソート
spell	英文スペル・チェッカー
style	文書の文体チェック
sum	チェック・サムの計算
split	ファイルの分割
dd	ファイルの変換とコピー
tr	文字の置換
find	ファイルの検出
join	関係データベース操作
ディレクトリ操作	
ls	ディレクトリ表示
cd	ディレクトリの移動
mkdir	ディレクトリ作成
rmdir	ディレクトリ削除
エディタ	
ed	行エディタ
vi	スクリーン・エディタ
sed	ストリーム・エディタ
コマンド・プロセッサ（シェル）	
sh	標準シェル
csh	C シェル
文書整形	
nroff	文書の整形（タイプライタ用）
troff	写植機用文書の整形
ditroff	出力装置に依存しない文書の整形
tbl	作表用プリプロセッサ
pic	作図用プリプロセッサ
eqn	数式作成用プリプロセッサ

開発ツール	
as	アセンブラ
ld	リンカ
nm	名前表の表示
cc	C コンパイラ
cb	C プログラム整形ツール
indent	C プログラム整形ツール（BSD）
lint	C プログラムの整合性チェック
ctags	C プログラムのタグ作成
error	エラー・メッセージの解析
prof	プログラム実行プロファイルの表示
make	プログラムの更新
f77	Fortran コンパイラ
ratfor	Ratfor プリプロセッサ
sccs	バージョン管理ツール（System V）
rcs	バージョン管理ツール（BSD）
lex	字句解析プログラムの生成
m4	マクロ・プロセッサ
yacc	コンパイラ・コンパイラ
adb	デバッガ
dbx	シンボリック・デバッガ（BSD）
sdb	シンボリック・デバッガ（System V）
awk	パターン処理言語
アーカイバ	
tar	テープや FD へのファイルのアーカイブ
uuencode	バイナリ・ファイルのテキスト化
uudecode	上記のファイルの復元
compress	ファイルの圧縮
uncompress	圧縮ファイルの復元
スケジュール管理	
at	日時指定でコマンドを実行
cal	カレンダ出力
calendar	予定表の検索
補助ツール	
dc	電卓
bc	任意精度の電卓（dc のフロント・エンド）
通信	
cu	ターミナル・エミュレータ（System V）
tip	ターミナル・エミュレータ（BSD）

三つ以上のコマンドがパイプで連結された場合も，同様にコマンドの数だけのプロセスを起動して並列に実行していくので，効率よくフィルタの処理がおこなわれる．その点ではUnixの環境になれてしまうと，MS-DOSのパイプの処理が貧弱で効率のわるいものにみえる場合がある．

FIG 図3 3

```
foo > tempfile
bar < tempfile
```

FIG 図4 4

```
int pid1, pid2;
int pf[2];        /* パイプ用のファイル・ディスクリプタ */

pipe(pf);         /* パイプを作成し、pf[1]を入口、pf[0]を出口とする */

if ((pid1 = fork()) == 0) {      /* 1番目の子プロセス */
    close(1);
    dup2(1, pf[1]);      /* 標準出力をパイプの入口に */
    close(pf[0]);
    close(pf[1]);
    execl("foo", "foo", 0);
}

if ((pid2 = fork()) == 0) {      /* 2番目の子プロセス */
    close(0);
    dup2(0, pf[0]);      /* 標準入力をパイプの出口に */
    close(pf[0]);
    close(pf[1]);
    execl("bar", "bar", 0);
}

close(pf[0]);
close(pf[1]);

while (wait(0) >= 0)     /* コマンド終了のまちあわせ */
    ;
```

GNUとソフトウェアツール

Unixが一般に普及することで，もともとはフリー・ソフトウェアにちかい感覚で流布していたUnixのソース・プログラムに，商品としての枷がはめられるようになってきた．それまでのUnixでは，特に大学などでソース・プログラムの修正や改良がおこなわれたり，あたらしいソフトウェア・ツールが開発されたりしてきた．その中でも有名なものがカリフォルニア大学のバークレイ校からリリースされた，BSD（Berkley Software Distribution）である．

それに対してAT&Tが商品としてサポートしているUnixが，System Vとよばれるものである．現在ではこの二つのUnixの，統合化がおこなわれようとしている．

しかしUnixが商品としてライセンスされてくることで，*Software Tools*という本でしめされたようなかんがえかたの一部が，変化してきているのもたしかである．多分現在のベル研究所からは，そのような本がでてくることはむずかしいという気がする．

そのような傾向に対抗するという意味もあって出現したのが，Richard StallmanひきいるFSF（Free Software Foundation）によるGNU（GNU's not Unix）プロジェクトである．GNU Emacsという高機能のスクリーン・エディタ＋ソフトウェア環境を代表として，CおよびC++のコンパイラ，アセンブラ，デバッガ，各種のUnix互換ツールが，ソース・プログラムのかたちで自由に流通されている．近日中にはBSD互換のOSカーネルもリリースされるみこみであり，将来はUnixを完全にGNUのソフトウェアでおきかえることも可能になりそうである．

現在MS-DOS上で利用できるUnix互換のソフトウェア・ツールには，GNUのツールを移植したものがかなりあり，DOS-Extenderを利用することで，コンパイラやEmacsなどの巨大なソフトウェアも動作する環境がととのえられつつある．その意味ではUnixマシンとパーソナル・コンピュータの間で，ソフトウェア・ツールの充実度の差はうまりつつあるともいえる．

ツールの達人

ツールに甘えず，使いこなす

株式会社マイクロフト 平田祥司氏――に聞く

平田氏は，ハードがらみの組み込み型プログラムから事務系ソフトの開発まで，幅広いプログラミングを行っているプログラマである．使用するマシンもPC-9801からIBM PC/AT互換機，さらにUNIXワークステーションまでと多彩である．
「便利なツールに甘えてはいけない」とする平田氏に，自身が利用しているプログラミングツールについて語ってもらった．

ハードウェア

98，IBM PC/AT互換機，ワークステーションをマルチに活用

メインマシンとしては，130MBのHDDと10MBのRAMをつけたPC-9801 RA21，ほかにIBM PC/AT互換機とSunのワークステーションを使用しています．これらにはイーサネット用のボードを入れて，LANを構築しています．ネットワークOSはNetware，サーバマシンにはIBM PC/AT互換機を使用しています．

Sunのワークステーションには1.6GBのHDDがついていますが，これはバックアップ用にしか使っていません．ほんとうはもっと活用したいのですが，なかなかSunでやる仕事がこないものですから．

客先でプログラムの修正を行うときのためにノート型パソコンも使っています．やはり，自分のプログラミング環境がないと不便ですから．

コンパイラ

Borland C++がメイン，MS-DOSでアセンブラは使わない!

メインで使用しているコンパイラは，Borland C++（以下BC++）．Cの仕事は，特別に客先から指定がない限りこれだけ．IBM版と98版の両方を用意しています．

お客さんによってはMS-Cを指定することもあるので，そのときだけはMS-Cを使います．ただし，最近ではほぼBC++が100%です．

CPUボードなどの組み込みプログラミングを行うとき以外は，MS-DOS上でアセンブラを使うことはありません．また，アセンブラを使わなくてすむようにプログラミングしています．

仕様の打ち合わせから決定まで

ファックスボードとワープロを活用

プログラムの打ち合わせが始まると，まずファックスボードが活躍します．HALファックスを使っているのですが，ドキュメントや仕様書のやりとりはこれでやります．

ドキュメントの作成は，松Ver.5を使っています．この段階ではとくにツールは使いません．

プロトタイピング

画面作成ライブラリを用いてプロトタイプを作成

事務系のソフトを作成する場合，お客さんからもっとも修正の要求が多いのが画面まわりです．そこで，まずソフトの画面から作る必要があります．

動きはしないけれども一通り画面が見れるようなものを作って，お客さんからOKをもらうわけです．このとき，IBM PCではCodeBase4[*1]というライブラリを使っています．CodeBase4を使ってメニューや入力用の画面を作り，BC++でコンパイルしてお客さんに見せるわけです．

ここでお客さんのOKがでると，こんどはCodeBase4のC++版のクラスライブラリを使ってプログラムを作成します．

98にはあまり使いやすい画面作成ソフトがないので，自作のライブラリを使ってプログラミングしています．

＊1 CodeBase4
IBM PC用のdBASE互換アプリケーションを作成するための，C／C++用のライブラリ．
連絡先：Sequiter Software Inc.
P.O. Box 5659,Station L,Edmonton,Alberta,Canada T6C 4GL

ソースの編集

究極のエディタはやはりQEdit

エディタは，IBM PC/AT互換機ではMifesかQEdit[*2]を使っています．98ではMifesです．キーアサインを同じにして，両方チャンポンで使っています．できればQEditで統一したいのですが，98用のQEditがないのでしかたなくMifesを使用しているわけです．

Mifesの欠点としては，ネットワークドライブにアクセスできないこと，検索にしても置換にしてもファイルを読み込んで編集できる状態になるまでにしても，とにかく遅いということです．QEditは軽くて速いですし，C言語のインデントを自動的にやってくれる，ここが気にいっています．もちろん，ネットワークにも対応しています．

スピードではVzエディタという選択もできるのですが，Vzエディタは1つのファイルを複数ウィンドウでアクセスできないので使いません．同時に1つのファイルを複数箇所修正したいとき，たとえばCのヘッダ宣言・プロトタイプ宣言を書いている部分と実際に関数を書いている部分の2箇所を表示して同時に修正したいときなどが，プログラム作成ではよくあります．QEditは完全にそれができます．Mifesもスピードが遅いことを別にすれば，なんとかできます．Vzはまったくできません．BriefはQEditよりもさらに1つ進んでいて，下のウィンドウを変更すると上のウィンドウも同時に変わっていきますね．QEditではあとでまとめて変更されます．一時期Briefを使おうとも思ったんですが，あのキー配置に慣れなくてあきらめました．

＊2 QEdit（quick editor）
IBM PC用のエディタ．シェアウェアとしてパソコン通信などを通じて配布されている．レジスト料金は，$54.95（マニュアル付）．
連絡先：SemWare(R)
4343 Shallowford Road, Suite C3A
Marietta, GA 30062-5022
VOICE:(404)641-9002
FAX:(404)640-6213
BBS:(404)641-8968

コンパイル デバッグ

統合環境もデバッガも使わない，甘えは禁物！

make

makeは，最近はBC++付属のmakeも使うようになりましたが，基本的にはOPUS MAKE[*3]を使っています．これはエージーテックが輸入しているmakeで，強力なマクロが備わっているのが魅力です．

BC++付属のmakeの優れている点は，複数のソースファイルをまとめてBCCに渡してくれることです．通常のmakeでは，ソースファイルが10個あると10回BCCをロードするわけです．しかし，BC++付属のmakeはターゲットをずっと見ていって一気にBCCにソースファイルを渡すことができます．こうするとBCCを1回しか読み込まないので，その分かなりコンパイル時間が短くなります．プログラミングをはじめると，100行／200行程度の細かいファイルがたくさんできるので，この機能はありがたいですね．

リンカは，BC++付属のものを使っています．とくにリンカには凝りません．

＊3 OPUS MAKE
米OPUS Software社が開発したUNIXのMAKEと互換性のある強力なMAKEユーティリティ．
問い合わせ先：（株）エージーテック
TEL 052-951-2706

デバッガ

デバッガはあまり使いません．ぼくのプログラミングスタイルは，リストを何回もコンパイルしては少しずつ直してあたりをつけていくというものです．普通の人のようにデバッガでずっと追いかけていってあたりをつけて，ここだなとソースを直していっきにコンパイルというのではなく，カチャカ

チャと直してはコンパイルしてみて「あれおかしいぞ,動かねぇぞ.」,またカチャカチャと直して「仕方がねえな,ここまではきているからprintf()入れてみよ」という感じです.机上デバッグといえばかっこいいんですが,あたるを幸い手あたりしだいにやっているわけです.

組み込み系の仕事の場合にはまともなデバッガがないことが多いですから,このような原始的な方法を取らざるを得ないということもあります.

いつデバッガのない環境でデバッグしなければいけないかわかりませんから,普段からデバッガには頼らないことにしています.甘えちゃいけません！

統合環境も同様の理由で使いません.あれに体が慣れてしまうのが恐いのです.ある日突然,edlinしかないような環境にいっても仕事ができるよう,体を鍛えておかなければ…….

awk/sed/grep/edlin

その場限りのツールを作るのにawkを利用しています.これは,MKSのawk*4です.MKSのawkのほうが,GNUのものより数倍速いので気に入っています.

awkを使い始めてから,ツールを作ったり,探したりしなくなりました.awkでできるというか,むりやりやってしまいますから.

sedはGNUのものを使っています.ただし,sedでできることは基本的にawkでできますからあまりsedを使うことはありません.とくにメイクファイルの中に書かなければいけないスクリプトのようなものはsedを使って書いています.

複雑なメイクファイルになると,まずawkで処理して,次に処理速度の速いsedを通して,さらにedlinを通して……,なんてやっているときがあります.

grepは,BC++に付属しているものを使っています.これは大文字・小文字をデフォルトで無視してくれるのが気に入っています.

メイクファイルなどで文字列の連続置換をやるときには,よくedlinをフィルタとして使います.たとえば,1つのソースの中の100種類の単語を書き換えたいという場合,sedではズラズラと書いていかなければなりませんが,edlinならば「r/単語2個」というファイルを作っておけばいいだけですから.

昔,Cのソースを漢字で書きたいというわがままなユーザーがいまして,このときはedlinを駆使して,「printf()」を「印刷」とか書くことができるようなものを作りました(笑).

＊4 MKS awk
awkは,AT&Tで開発されたファイル処理言語である.ちょっとしたツールを作ったり,試験的なプログラムの作成に威力を発揮する.MKS awkは,カナダのMortice Kern Systems, Inc.が開発・発売しているawkインタプリタである.日本では,パーソナルメディア株式会社からPC-9801版／IBM PC版がMKSツールキットとして発売されている.
問い合わせ先：パーソナルメディア(株)
TEL 03-3490-8841

ドキュメンテーション
ソースの整形にindent,いいアウトラインプロセッサが欲しい

ドキュメント作成には苦しんでいます.昔はPC-Outline*5というアウトラインプロセッサを使っていたのですが,漢字がうまく通らないことが多く,問題があったのでいまは松を使っています.まぁ,書きにくくてしょうがないですね.

アウトラインプロセッサまたはアイデアプロセッサのいいのがあれば,ぜひ欲しいですね.管理工学さん,なんとかしてくれませんかね.

客先にソースを提出しなければいけない場合には,indentというソースの形を整えるソフトを使っています.

ただ,これにはバグがあって,c=*pとかp=&aをc*=pやp&=aに直してしまうんです.昔のCの文法ではこの技法が許されているので,これがまだ残っているんだと思うんですが,ポインタ指定の部分が破壊されてしまうので,あとで修正する必要があります.誰かあのバグとってくれないかなぁ.最近のGNUのテープをみてもバージョンも変わっていないし,みんな使っていないのかな？

最近の仕事では,ドキュメントを印刷したものとは別にファイルでくれということが多いので,TeXやroffなどは使いません.

日本語入力フロントプロセッサは,やはりFIXERを使っています.これは,友人と「FIXERを守る会」を結成しているくらい気に入っています.皆さんにもぜひ使っていただきたいですね.

＊5 PC-Outline
英文アウトラインプロセッサ.シェアウェアとして,パソコン通信などを通じて配布されている.レジスト料金は,$89.95(ディスク,マニュアル付).psh氏が作成したsim(IBM PC BIOSシミュレータ)を使用することで,制限付きではあるもののPC-9801上で使用可能になる.
問い合わせ先：Brown Bag Software
2155 South Bascom Ave
Campbell CA 95008
または,
File #41719 Box 60000
San Francisco CA 94160-21719
PHONE:(408)559-4545
BBS:(408)371-7654

いまやスクリーンエディタはソースプログラムの入力・編集になくてはならないプログラマ必携ツールである．統合環境を備えた言語処理系が増えて，コンパイラを買えばスクリーンエディタが付いてくるようになった昨今でも，1度手に馴染んだエディタからはそう簡単には離れられないという人が多いと聞く．その理由のひとつには，自分の使うエディタをより使いやすく，便利なものへと改良していくことができる，強力なマクロ記述能力と，カスタマイズ機能があげられるだろう．本章ではプログラマに人気の高い3つのフルスクリーンエディタ――MIFES，VZエディタ，Briefを取り上げ，各エディタのマクロの達人にプログラミングの役に立つ鮮やかなマクロの使いこなしぶりを披露していただいた．

特集
プログラミングツール大集合

マクロの達人

PROGRAMMING
TOOL

マクロの達人 MIFES

「外部」と「英字」，2種類のマクロをもつ

志村　拓

なぜマクロなのか

Turbo PASCALがコンパイラにエディタを組み込んで以来，パソコンでも統合化環境としてのコンパイラが一般的になりつつある．ソースコードを書くところから，コンパイルはもちろんのことデバッギングに至るまでを1つの環境で行うことができるのだから，以前にくらべてプログラムの開発効率がいいことはいうまでもない．こうなると，単体としてのエディタはその存在価値が問われてしまう．バッチプログラムや，統合化環境の用意されていない処理系のため，またワープロの代用品として，その価値を見いだしていくしかないのだろうか？

筆者はUNIXワークステーションのプログラマであるので，従来通りエディタでソースプログラムを書き，makeコマンドでコンパイルを行う．統合化環境を使用するのは，そのデバッガを利用する場合だけだ．「なぜ今時統合環境のエディタを使わないで単体のエディタを使用するのか？」という問もあるだろう．しかし，手に馴染んだエディタとは，それを触っているだけで想像力を掻き立てるものだし，自

LIST 1　リスト1　MIFESをコンフィグレーションするためのmakefile

```
# Makefile for MIFES Customize
# Copyright 1991 (C) by Taku-SHIMURA
#
# Makefile for ASCII MS-DOS Tools' make.exe
# SHeader: makefile.v  1.1  91/10/26 01:10:00  taku  Exp $
#

MILC    =       milc
MIKEY   =       mikey
MIORG   =       mifesorg.exe
MIFILE  =       mifes.exe
MIDEST  =       a:¥usr¥bin
MIPATH  =       a:¥usr¥bin¥mifes.exe
DEF     =       taku.def

DEFS    =       emac.def  ¥
                etc.def   ¥
                menu.def  ¥
                color.def ¥
                guid.def  ¥
                key.def

all:            $(MIPATH)

clean:
                @del $(DEF)
                @del $(MIFILE)

co:             $(DEFS)

$(MIPATH):      $(DEF) $(MIFILE)
                @echo Customize $(MIFILE) by $(DEF) ...
                @$(MIKEY) $(DEF) $(MIFILE)
                @echo done.
                @echo Install $(MIFILE) to $(MIDEST) ...
                @copy $(MIFILE) $(MIDEST) /v > nul
                @echo done.

$(MIFILE):
                @echo Dupricate $(MIORG) to $(MIFILE) ...
                @copy $(MIORG) $(MIFILE) > nul
                @echo done.

$(DEF):         $(DEFS)
                @echo Binding definition files ...
                @cat $(DEFS) > $(DEF)
                @echo done.

$(DEFS):
                @co $@
```


おもなUNIXコマンド
ls　ファイル・ディレクトリ名と属性の表示

LIST リスト2 行移動英字マクロ 2

```
*A=行移動(下方向)
        #75#Down#Paste_L#Up
*B=行移動(上方向)
        #75#Up#Paste_L#Up
```

LIST リスト3 数値変換英字マクロ 3

```
*C=文字->16進コード変換
        input(1, "文字入力")
        @1 = @str1[0] & 0x1F
        printf("02x", @1)
*D=10進数値->2進コード変換
        input(1, "数値入力")
        @1 = atoi(1)
        printf("016b", @1)
```

分の思い通りに動くエディタでバリバリとコーディングすることは，一種の快感さえ感じるものなのだ*1.

べつに統合化環境を否定しているわけではない．冷めた目で見れば，単に統合化環境のエディタに移行できないでいる不器用な人間なのかもしれない．しかし，プロのプログラマの中には，こうしたスタイルを取っている人が少なからずいることもたしかだ．要するに手に馴染んだツールというものは，なかなか手放すことができないものなのだ．また，処理系によって統合化環境エディタのキーアサインや機能が変わってしまい，思考の中断を余儀なくされてしまう点を嫌う結果でもあるわけだ．最近のワープロが日本語FEPを特定せずに動くように，言語処理系の統合化環境もエディタのAPIを定めて，その規格に合っていれば，どのエディタでも使用できるようになれば，最高なのだが．

さて，前置きはこれくらいにして，筆者が日ごろプログラミングの際に使用しているMIFESのマクロを紹介することにする．

＊1 この時点では，その後に起こる面倒なデバッグ作業のことなどは頭の中にこれっぽっちもないので，まさにプログラマとして至福の瞬間といえるのかもしれない．

MIFESのマクロ

MIFESには，その編集機能をプログラマブルに定義して自動運転する「マクロ」と呼ばれる機能がある．マクロには，「外部マクロ」と「英字マクロ」の2種類がある．

外部マクロとは，その名が示す通りマクロプログラムが1つのファイルとして，エディタ自身の実行ファイルの外にあるタイプのものだ．これに対し，英字マクロはエディタ内部の「英字バッファ」に格納されたマクロプログラムであり，エディタ自身の実行ファイルの中に格納される．

英字マクロを作成するには，MIFESのコンフィグレーションファイル(通常はMIFES.DEF)の中の英字バッファにマクロプログラムを書く．次に，これをMIFESのコンフィグレーションプロセッサであるMIKEY.EXEにかけて，MIFESを再コンフィグレーションしてやればよい．これによって，英字マクロは，MIFESの実行ファイルであるMIFES.EXEの中に取り込まれることになる．ちなみに，この方法のほかにも，起動後に動的に英字バッファにキーボードマクロを登録するといった方法があるが，キーボードマクロでは制御構造が使えない点，プログラムを終了してしまうと，その内容が消えてしまう点から，あまり有効とはいえない．

外部マクロは，実行の際にマクロファイルをMIFESの外部マクロ用のバッファに読み込んでから実行するが，英字マクロはMIFESの実行ファイルの一部としてMIFES実行時にメモリに展開されているので，外部マクロにくらべて素早く実行できるし，実行したいマクロファイルが見つからないといったこともない．頻繁に使用する機能は，この英字マクロに登録しておきたい．

再カスタマイズ

さて，ここで本題から少し話がそれてしまうのだが，MIFESのコンフィグレーションについて書きたいと思う．皆さんはMIFESのコンフィグレーションを行ったことがあるだろうか？実はこれが結構うざったい作業なのだ．問題は，コンフィグレーションファイルであるMIKEY.DEFにある．これが長いファイルで*2，たとえばキーアサインを変更したいと思ってMIFESでMIKEY.DEFを開いても，その場所を探すのが面倒なのである．

筆者の場合，友人*3の影響で，コンフィグレーションファイルを，その内容ごとにいくつかのファイルに分割している．MIKEY.EXEでコンフィグレーションを行う際には，これらのファイルを1つにまとめるのだ．こうした作業は，makefileに記述してあるので，実際のコンフィグレーション作業

は，make〈CR〉一発ですんでしまう．

このようにファイルを分割することで，変更したい箇所を素早く修正することができるし，ほかの定義の部分を誤って破壊してしまうこともなくなる．さらに各々のコンフィグレーションファイルをRCSによって管理することによって，コンフィグレーションファイルの細かなバージョン管理も可能だ．MIFESフェチには，この方法をぜひお勧めする．参考までに，筆者が使用しているmakefileをリスト1に載せておく．

＊2 MIFES Ver.5.Xから，画面の色などコンフィグレーションできる内容が増えたため，それまでのものよりさらに長いファイルになっている．

＊3 友人とは「MIFES Ver.5を256倍使うための本」や「VZを256倍使うための本」を筆者と一緒に書いた可部正三氏である．

小意気な英字マクロ

さて，本題に戻ることにしょう．先にも説明した通り，英字マクロは外部マクロにくらべ，素早く，しかも確実に動作する点から，頻繁に使用するマクロを組み込んでおきたい．外部マクロを使用したバキバキに凝ったマクロよりも，実際にプログラムをコーディングする際に影響を与えるのは，英字バッファに登録された,小さなマクロであることを忘れてはないらないわけだ．

とはいうものの，1つの英字バッファの大きさは，たかだか94バイトしかないので，本当にチョっとしたものしか作れない．複数の英字マクロをチェーンして使用すれば，何とか大きなマクロも作れるが，ここは小さくても有用なマクロを多数登録しておきたい．ということで，筆者のMIFESの英字マクロに登録されているものを2～3紹介することにしよう．

行移動

まずはじめに紹介するのは，行移動マクロだ(リスト2)．これは，カーソルのある行をその上の行や下の行と入れ換えるものだ．このマクロのポイントは，カーソルが入れ換えた行に追従して動くので，連続して使用すると，まるでその行を摘んで動かしているように動作するところだ．例えば，forループから余分な命令をそのループから外に出したり，命令の順番を変更したりといった場合に便利だ．

次は簡単な数値変換だ．1つは入力した文字のコードを16進数で現在のカーソル位置に書き込むマクロだ．このマクロは，漢字もサーポートしており，シフトJISコードを出力する．

10進数→2進数変換

もう1つは，入力した10進数を16ビットの2進数に変換して現在のカーソル位置に書き込むマクロだ．16進数，8進数，2進数の変換は比較的簡単に暗算で行うことができるが，10進数は少々面倒だ．このマクロを使えばMIFESにその作業を委せることができるわけだ．「そんなものは，ヘキサ電卓でやればいい」という向きもあるだろうが，電卓というものは，不思議なもので必要な時に限ってどこかにいなくなってしまう(筆者だけか？)．

MIFES Ver.5.xから，マクロで簡単なprintfモドキが使えるようになったので，このあたりの処理は非常に簡単に記述できる．

行頭の{へジャンプ

次は，行頭の'{'を検索してジャンプするマクロだ(リスト4)．これはviの'[['と']]'の動作を真似たものだ．こんな簡単な英字マクロもC言語でプログラムしている場合は，非常に有効だ．関数宣言の'{'の位置を必ず行頭に書くようにしていれば，関数単位のカーソルジャンプができるわけだ．

キーワードへジャンプ

カーソルをジャンプさせるといえば，タグジャンプを忘れてはならない．MIFESのタグジャンプは，ファイル名と行番号が書かれている行にカーソ

LIST リスト4　関数ジャンプ1英字マクロ　4

```
*E=前方関数ジャンプ
        #Find_F {'^¯¥{^M}
*F=後方関数ジャンプ
        #Find_B {'^¯¥{^M}
```

LIST リスト5　関数ジャンプ2英字マクロ　5

```
*G=タグジャンプ
        #Right#BackWard
        getword(1)
        @2 = open("TAGS")
        if 35 == @2 then @2 = text(30) & 0xFF
        if @2 < 15
                select(@2)
                #Top
                search(1, 0)
                #TagJp
        endif
```


おもなUNIXコマンド
man　ヘルプの表示

ルを合わせてタグジャンプ機能（通常は[f･10]にアサインされている）を実行することによって，その行に書かれたファイルを開き，その行にカーソルをジャンプさせるといったものだ．

これは，コンパイルでエラーがおきた場合に，その内容をファイルにリダイレクトして，エラー箇所にカーソルを飛ばすといった目的には非常に便利なのだが，ソースプログラムをバリバリコーディングする際にはほとんど使用できない．

そこで，viの‘Ctrl+]’と同じように，ソースプログラム上のキーワード（C言語ではおもに関数名）にカーソルを合わせて実行すると，そのキーワードが定義されているファイルを開き，そのキーワードが定義されている場所にカーソルを飛ばすようにしたのが，次のマクロだ．このマクロを使用するには，カレントディレクトリにMITAGS.EXEを使用してキーワードのテーブルファイル‘TAGS’を作成しておく必要がある．

これと同等（正確にはこれよりもまともな）のマクロが，MIFESのシステムディスクの中のMACRO.SRCにfntagjpという名前で入っている．しかし，残念ながらこれは外部マクロなのだ．本マクロは英字マクロのサイズの制限からfntagjpにくらべれば，エラートラップも甘く，メッセージの表示もないが，いつでも確実に素早くこの機能を利用できる．最初にも説明した通り，このような頻繁に使用する機能は英字マクロとして実現したいわけだ．

サイズの制限といえば，このマクロのサイズは91バイトで，ギリギリのサイズだ．最初にこのマクロを書いた時は128バイトもあった．当然英字マクロとしては登録できない．英字マクロに登録して，MIKEY.EXEにかけてサイズがオーバーして，コンフィグレーションが中断された場合は，その英字マクロを外部マクロとして作成しなおし，ソースコードを削りながらMILC.EXEでコンパイルするといった操作を繰り返し，94バイト以内に収まるまでガンバルのだ．

この限られた狭い空間の中に，いかにして納得できる機能を押し込むかを思案することの楽しみというのは，マイコンプログラマの特権だったはずだ．このごろはメモリのサイズもメガ単位が当たり前となり，プログラムもメモリにモノをいわせて，自由奔放に書くことが普通のようだが，ここで一昔前のプログラマの苦しみと楽しみを味わってみるというのも一興かもしれない．

ここで紹介した英字マクロは，[ESC]+?といった2ストロークのキー操作ではなく，1ストロークのキー操作にアサインして使用したい．1ストロークのキー操作のエントリーは少ないように思われがちだが，[SHIFT]+?を合わせるとそれほどでもないし，1ストロークのキー操作にアサインされている機能の中にもほとんど使用していないようなものもあるので，そのあたりをバシバシと変更してしまえばよいのだ．

外部マクロ

外部マクロは，最大1,500バイトの大きなマクロを作成することができるので，比較的凝った機能をマクロによって実現することができる．しかし，大きなマクロを組めるからといって，何でもかんでもマクロで実現しようと真剣に思ってはならない（冗談ならよい）．MIFESのマクロはそれほど速くないので，たとえばマクロでソートを行うようなプログラムを組んでも，あまり実用的ではないからだ．MIFESのエディタとしての機能を活かしたテキスト処理に的を絞って，実用価値のあるマクロを作成するようにしたい．

全テキストを対象にした置換

さて，最近作ったもので，よく使っている外部マクロに全テキスト置換マクロがある．これはgrepの置換版*4のようなもので，MIFESでオープンしているすべてのテキストに対して置換操作を行うマクロだ（リスト6）．

プログラム開発当初に適当に付けた関数名が，後からどうしても気に入らなくて変更したくなることはよくある（これも筆者だけ？）．このマクロは，こんな関数名や外部変数名などのプログラム全域にわたるキーワードを変更する場合に使用すると便利だ．MIFES Ver.5.xから検索・置換に正規表現が使用できるようになったので，かなり正確に指定した変数名や関数名だけを置換することができる．

＊4 'g/RE/p'ならぬ's/RE/RE/g'といったところか？

カウントアップ付き行の2重化

次に紹介するのは，自動番号カウントアップ付き行2重化マクロだ*5．これは，行の二重化（通常は[SHIFT]+[f10]にアサインされている）に，2重化した行の最初の数値をカウントアップする機能を付加したものだ．

つまり，2重化した行の先頭から数値文字列を探し，その数値文字列を1

加えたものに変換するのだ．例えば，プログラム中で配列に数値を代入する場合などや，データファイルの頭にレコード番号が付いているような場合に，このマクロが威力を発揮するのだ．このマクロは，本来なら英字マクロとして登録したいところなのだが，どうガンバってみても97バイト以下にはならないので，外部マクロとして実現した．こうしたマクロは「外部マクロの実行」メニューから呼ぶのではなく，キーにアサインして使用したい．MIFES Ver.5.xから，すべてのキー操作に外部マクロをアサインすることができるようになって便利になったのだから．

＊5 なんか，このあたりまでくると，ネーミング自体がマッドサイエンティスト「みのり」ちゃんの発明品のようで，非常にきな臭い．

Cの関数を切り出す

さて，最後にもう1つ，C言語でプログラミングする初期の段階で，筆者がよく使う外部マクロを紹介しよう．

筆者のプログラミングスタイルは，トップダウンでもボトムアップでもない．思いつくがままに，いきあたりばったりでプログラミングしてしまう，強いていえば，ミドルアップダウンスタイルを取っている．＊6

こんなふうに，いい加減にプログラムを作っているわけだから，最初からモジュールを設計してファイルを分割してソースプログラムを記述するといったことはない(キッパリ！)．最初は簡単なテストルーチンを書き，動かしてみてうまく動くようならば，さらにそれに改良，機能アップを施していくといったことを繰り返し行うわけである．ソースファイルが1つならmakefileを書く必要もないし，コマンドラインでチョコチョコっとコンパイラを動かすだけで実行ファイルができてしまうのだから，面倒がなくてよいのだ．

そうこうするうちに，ふと気がつくと，test.cというファイルは，関数が十数個も含まれる巨大なものになってしまうわけだ．さすがに，こういった状態だとコンパイルにも時間がかかるようになるし，ソースプログラムの目的の位置にカーソルを飛ばすことも面倒になってくる．だいいち，このままの状態では，恥ずかしくて人にも見せられない．

そこで，登場するのがリスト8のC言語関数切り出しマクロなのだ．このマクロは，現在カーソルのある位置の下にある関数＊7を，指定したファイルに切り出すものだ．といっても，ただ切り出すだけでは，あまりにも能がない．このマクロは，指定したファイルが新規のファイルであった場合，カレントテキストの先頭から最初の関数の前までの内容も指定したテキストの先頭にコピーしてくれるのだ．つまり，プログラムの最初に書かれたコメントやヘッダー，ディフィニッションシンボルもちゃんとコピーするのだ．

このマクロは，関数の切り出しをいい加減に行っているので，このマクロを適用するC言語のソースプログラムは必ず次のことを守ってほしい．1つは，関数の内容を括る‘{’と‘}’は行頭

LIST　リスト6　全テキスト置換　6

```
*sREg   全テキスト置換
;s/RE/RE/g  Version 1.0 (for MIFES v5.1)
;Copyright 1991 (C) by Taku-SHIMURA
;
;
; macro directive:
[#82    CONTREPL]
;
; variable list:
@1   = text(30) & 0xFF                              ;カレントテキスト番号
@2   = @num                                          ;カレント行号
;@3  =                                               ;メニュー番号
;@4  =                                               ;テキスト番号

        stopoff()

        setmenu(0, "文字列の全ファイル連続置換")          ;メニューの設定
        setmenu(1, "Replace Query    :連続置換：確認あり")
        setmenu(2, "Replace All      :連続置換：確認なし")
        setmenu(3, "Replace Continue:前回の連続置換の再実行 ")

        while 1                                      ;メインループ
                @3 = selmenu(3, 40) - 1              ;メニューの選択
                if 0xFFFE <= @3 then exit

                if 2 == @3                           ;置換の再実行
                        #ContRepl
                else                                 ;置換文字列の取得
                        if 0xFFFE <= input(1, " 旧文字列の入力") - 1 then exit
                        search(1, 2)
                        if 0xFFFE <= input(2, " 新文字列の入力") - 1 then exit
                        search(2, 2)
                        #Top
                        if @3 then areplace($FFFF)       ;置換
                        else       qreplace($FFFF)
                endif

                @4 = 15                              ;全テキストを置換する．
                while @4--
                        if @1 ==  @4  || select(@4) then continue
                        #ContRepl                    ;置換の再実行
                wend

                select(@1)                           ;元の状態に戻す．
                jump(@2)
        wend
```

LIST　リスト7　自動番号カウントアップ付き行二重化マクロ　7

```
*nDup   番号インクリメント機能付き行複写
;nDup  Version 1.0 (for MIFES v5.1)
;Copyright 1991 (C) by Taku-SHIMURA
;
; macro directive:
[#49    DUPLINE]
;
; variable list:
;@1                                                  ;ワーク

        #DupLine #Down #LeftEdge
        while (@code < '0' || '9' < @code) && @code < 0xFFFD
                #Right
        wend
        if 0xFFFD <= @code then exit
        @1 = 0
        while '0' <= @code && @code <= '9'
                @str1[@1++] = @code
                #Del
        wend
        @str1[@1] = 0
        @1 = atoi(1) + 1
        printf("d", @1)
```


おもなUNIXコマンド
mount　ファイルシステムのマウント

に書き，それ以外の‘{’と‘}’は絶対に行頭に書かないこと．まぁ，これはふつうそう書いていると思うが，構造体や共用体の宣言にも気をつけてほしい．もう１つは，関数の前と後ろには必ず空行を１つ入れてほしいということだ．コメントを関数の前に，その関数に関するコメントを書く場合は，そのコメントと関数の間に空行を入れずに，コメントの前に空行を挿入する．とくに，最後の関数の後ろにも空行が必要な点に注意してほしい．

このマクロを作るまでは，巨大なtest.cファイルを分解するに，結構な労力を消費していたが，今ではチョチョイのチョイでできるようになったため，いまだにミドルアップダウンのプログラミングが治らないのはいうまでもない．

＊6 情けないプログラマなわけだ．前出の友人に，「志村さんのプログラムはおおらかで，細かいことにとらわれないところが凄い」とお誉め(?)の言葉をいただいたことがある．

＊7 カーソルは，必ず関数の中ではなく，外に置いておくことに注意してほしい．もし，関数宣言の中にカーソルがあると，その関数の途中から，関数の終りまでを切り出してしまう．

その他のユーティリティ

あなたが，もしMIFESの正規ユーザなら，メガソフト社にユーティリティープログラムを入手するために連絡することをお勧めする．実費にて，メガソフト社から提供されるディスクには，最新版のサンプルリリースであるMIFESとともに，いくつかの有用なユーティリティプログラムが入っているからだ．

その中でもとくに秀逸なのが，筆者が最近気に入っているバイナリエディタ「MIBE.EXE」だ．これはMIFESのコンパクトモデルである「MINI.EXE」の亜流バージョンで，編集ファイルをヘキサダンプで表示し，バイナリデータなどを編集するための特殊なエディタだ(写真１)．

MIBEは，16進数や文字列によるファイルの編集はもちろんのこと，指定文字列や指定ビット長の数値による検索や，カーソル位置のデータに対する演算などといったことまでできる．さらに現在のカーソル位置の数値を文字，バイトデータ，ワードデータの形で表示したり，行番号の代わりにチェックサムを表示することができる．

これがあれば，バイナリ型データファイルの操作や，実行ファイルのパッチ当てが楽しくなることうけあいなのだ．

LIST 8 リスト8　C言語関数切り出しマクロ

```
*cCutOut        C言語関数切り出しマクロ
;cCutOut  Version 1.0 (for MIFES v5.1)
;Copyright 1991 (C) by Taku-SHIMURA
;
; macro directive:
;
; variable list:
@1 = text(30) & 0xFF                                ;カレントテキスト番号
;@2                                                 ;出力テキスト番号
@3 = @num                                           ;カレント行番号

        if 0xFFFD <= (finput(1, "ファイル名を入力してください") - 1) then exit
        @2 = open(1)
        if 34 == @2
                @2 = text(30) & 0xFF
                select(@1)
                #Top #Sel_L
                #Find_P {'^'¥{^M}
                #Find_B {'^'S^M}
                #Copy
                jump(@3)
                select(@2)
                #Paste_L
        else if 35 == @2
                @2 = text(30) & 0xFF
        else if 30 <= @2
                message("指定したファイルが開けません")
                exitm()
        endif
        select(@2)
        #Bottom
        select(@1)

        #Find_P {'[^' '1]^M}
        #Sel_L
        #Find_P {'^'¥^@^M}
        #Find_P {'^'S^M}
        #Down
        #Cut

        select(@2)
        {^M^M}
        #Paste_L
        #S_Q
```

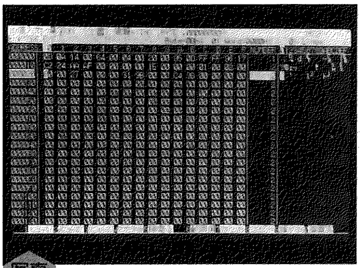


写真1 MIBE.EXE

MIFES

●製品名：MIFES Ver.5.0
●価格：38,000円（PC-9800シリーズ用，J-3100シリーズ用，PS/55用，DOS/V・AX用）
48,000円（FM-R/Panacom M用，B16/B32シリーズ用，if800EX/RX/BS21用）
●問い合わせ先：メガソフト（株）
TEL 06-386-2043

マクロの達人

VZエディタ

制御構文をもつ強力なマクロ

大野元久

VZエディタには，強力なマクロ機能が組み込まれていることはもう皆さんご承知でしょう．このマクロは，記号ばかりが並ぶので読みにくいといわれますが，実は書きやすいマクロでもあるのです．プログラマの皆さんが，VZのマクロ機能を使わずにいるなんてもったいないことです．ぜひ，VZのマクロを使ってより快適なプログラミング環境を整えてみてください．なお，ここでは説明のために.DEFファイルとしてVZ.DEFファイルを使い，プログラムとしてはC言語を取り上げます．

キーボードマクロ

VZのマクロ機能をもっとも簡単に体感できるのはキーボードマクロでしょう．プログラムを書いていて何度も同じことをしなければならなくなることはよくあるものです．たとえば，リスト1のプロトタイプ宣言から関数名だけを取り出すことを考えてみましょう（あらかじめ，このリストを入力するか，付録ディスクからコピーして用意しておいてください）．

どのプロトタイプ宣言も関数名の後ろに‘(’がついていますから，これをもとにマクロを定義してやればよさそうです．まず，[CTRL]＋[ROLLDOWN]でカーソルをテキストの先頭に移動させます．[CTRL]＋[Q]，[F]で検索文字列に‘(’を入力すると，ページングモードが検索モードになります（左上にSと表示される）．[CTRL]＋[_]を押すとキーボードマクロの登録がはじまります．ここでは表1のように入力してください．

最初の関数名（ここでは，_fstrcat）だけを取り出せたと同時に，[CTRL]＋[O]にプロトタイプ宣言から関数名を取り出すためのマクロが定義されました．あとは，[CTRL]＋[O]を何回か押すだけでこのファイルの中のすべての関数名が取り出せます（簡単でしょう？）．最後に[SHIFT]＋[INS]を押して，ページングモードを通常（P）に戻しておきましょう．

繰り返し処理を簡単に行うためには，[CTRL]＋[O]のような１回だけの

LIST　リスト1　sample.h　1

```
char far * far _fstrcat(char far *dest, const char far *src);
char far * far _fstrchr(const char far *str, int c);
int far _fstrcmp(const char far *s1, const char far *s2);
char far * far _fstrcpy(char far *dest, const char far *src);
char far * far _fstrdup(const char far *str);
size_t far _fstrlen(const char far *str);

void far * far _fmemchr(const void far *str, int c, size_t n);
int far _fmemcmp(const void far *s1, const void far *s2, size_t n);
void far * far _fmemcpy(void far *dest, const void far *src, size_t n);
void far * far _fmemmove(void far,*dest, const void far *src, size_t n);
void far * far _fmemset(void far *s, int c, size_t n);

```

CHART 表1 1

[CTRL]-[_]	マクロ定義の開始
[CTRL]-[C]	関数名の直後の'('を検索
[CTRL]-[A]	1ワード前（関数名の先頭）に移動
[CTRL]-[BS]	行頭まで（関数の戻り型）を削除
[CTRL]-[C]	関数名の直後の'('を検索
[CTRL]-[DEL]	行末まで（関数引数）を削除
[CTRL]-[O]	マクロの登録

キーストロークを空けておくことが望ましいでしょう．1キーストロークであれば，キーリピートが利用できるからです．1ストロークキーが空いていなければ，[CTRL]+[Q]，[G]のような2ストロークキーに割り当てることもできます．その場合は，1回マクロを実行してから[CTRL]+[[]（コマンドの再実行）を使ってキーリピートさせてください．

.DEFファイルのカスタマイズ

マクロとは繰り返し行われる処理を簡単にしようというものです．先に示したように，同じことを繰り返す処理があれば，それを形式化しマクロで実現できないかどうか考えてみるようにしましょう．考えすぎてかえって時間がかかってしまうようでは本末転倒ですが，プログラマのあなたなら処理を形式化することに抵抗はないはずです．

簡単な形式的処理なら，たとえばリスト2のようなマクロを記述できるでしょう．

このマクロを.DEFファイルのマクロの欄(＊Mではじまる一連のマクロ定義)に記述しておけば,空のファイルで[CTRL]+[K]，[M]と押してmain関数を記述するまでを自動的にテキストに挿入させることができます．

ここでは，ほとんどテキストに文字コードを挿入する機能しか使われていません．ダブルクォート（"）で囲まれた文字列を記述すると，その文字列がテキストに挿入されます．#mは改行をあらわし，#21はタブを，#16は行頭までの削除をあらわします．

繰り返し処理が単なるコマンド列の実行では済まなくなることがあります．リスト1を読み込みなおし，前述のマクロを定義した後，新たにMYMACRO.DEFというファイルをオープンしてください．ここで[CTRL]+[K]，[_]と押すと，定義したマクロが文字列としてMYMACRO.DEFのテキストに挿入されます．

挿入された文字列は，コメントやよけいなスペースはいっさい入りませんので少々わかりにくいかもしれません．ここで，#cは[CTRL]+[C]を，#aは[CTRL]+[A]を，#18は[CTRL]+[BS]を，#1は[CTRL]+[DEL]の機能をあらわします．ここで，#cが[CTRL]+[C]という入力に置き換えられるのではなく，機能番号36のページダウンが実行されることに注意してください．.DEFファイルを使ってカスタマイズすれば，[CTRL]+[C]を別の機能に割り当てることもできますが，マクロ中の#cはあくまでもページダウンをあらわします．

#+数字は，機能番号を意味します．機能番号の意味については，VZ.DEFのようなカスタマイズファイルを参考にしてください．

また，#+文字であらわされる記号を短縮シンボルと呼んでいますが，このうち#+英小文字であらわされるマクロ機能は，ほぼ[CTRL]+英字に相当すると覚えておくとよいでしょう．#+英大文字には，#B(Block，ブロック指定)，#M (Mark，カーソル位置の記憶)，#J (Jump，行番号ジャンプ)，#F (Find，検索文字列の設定)のように意味のついた記号が使われています．このほか，#，#<，#>，#_は，それぞれテキストの先頭，行頭，行末，テキストの末尾への移動を意味します．

LIST リスト2　空のファイルにmain関数を記述する 2

```
80 ^KM "main関数の作成"
        "#include <stdio.h>" #m
        #m
        "main(int argc, char **argv)" #m
        "{" #m
        #21 #m
        #16 "}" #m
        #e #e #d
```

LIST リスト3　テキストに挿入されたマクロ文字列 3

```
90 ^O "" #c#a#18#c#
```

実際にどのような記号が短縮シンボルに割り当てられているかを調べるには，[HELP]キーでヘルプを表示させるか，そこでキーを入力してみるとよいでしょう．

マクロの制御構文

さて，VZのマクロが強力といわれるのは，制御構文をもっているところにあります．このマクロにかざりをつけ，繰り返し文の形式にしてみましょう(リスト4)．

先頭の「＊ MACRO」と最後の「＊」は，このマクロをエディタ上でカスタマイズするための印です．この印をつけておけば，[CTRL]＋[K]，[^]と入力することでそのテキストをマクロとして取り込むことができます．

マクロの第1行目はマクロ番号，割り当てキー，タイトルをあらわしています．2行目の「?.」は，エディットモード以外ではこのマクロを終了させることを意味します．これを正確に記述するならば「(s)?.」となります．sという変数は，その値で次のシステムモードをあらわします．

LIST 4 リスト4　修正されたマクロ

```
* MACRO

90 ^O "関数名の取り出し"
    ?.              ; 編集モードでのみ有効
    #^ #<           ; ファイルの先頭へ移動
    mp[,            ; ページングモードの退避
    #F "(" #m       ; 検索文字列の設定
:A                  ; 繰返しラベル
    #c              ;  '(' を検索
    (r)?{mp], .}    ; 検索に失敗したら終了
    #a              ; 1ワード前（関数名の先頭）に移動
    #18             ; 行頭まで（関数の戻り型）を削除
    #c              ;  '(' を検索
    #l              ; 行末まで（関数引数）を削除
    >A              ; マクロの繰返し

*
```

CHART 2 表2

0	エディットモード
1	文字列入力モード
2	DOS のコマンドライン
3	ファイラー
4	メニューなど1文字入力モード

条件分岐について少し説明します．?は条件実行を意味し，直前の式の評価が0でなければ直後の文を実行します．式を評価するためには，その式をカッコでくくらなければなりません．逆に，カッコでくくらなければ式は評価されません（カッコでくくらない式は，「,」で終了させなければなりません）．ここで，()と?が別のものであることを覚えてください．

たとえば，「(a) a =0, ?# c」というマクロでは，aが評価されてから，aに0が代入され，先に評価されたaが0でなければ#cが実行されます．(a)の後ろにa =0があるからといって，#cが実行されないということはありません．また，評価値は?を判定したあとで真偽が反転します．「(a)?# s?# d」というマクロでは，(a)を評価し，評価値を反転します．先に評価した結果が真(非0)であれば#sを実行し，そうでなければ#sをスキップします．次に?に出会うと評価値が反転しているため先ほどと逆の条件で#dを処理するかしないかが決まります．こうして，if～then～elseという形式の構文が非常に単純に実現されているのです．

この動作を理解することは，VZのマクロで条件分岐を使う際に非常に重要です．実行される文の中に，式の評価が入っていれば，そのあとの?はその評価値だけに依存してしまいます．VZのマクロがifブロックをネストできない(しにくい)理由はこの単純化された処理のためです．

さて，もとのマクロに戻りましょう．「.」はマクロの終了を意味しますから，これは現在のモードがエディットモードでなければマクロを終了するという意味になります．「?.」と省略されていたのは，どのマクロもそれが呼び出される直前で(s)という値の評価が実行されているためです．多くのマクロはテキストを編集するときだけに有効なものですから，「?.」を入れておくようにしましょう．これがないと，行入力のときに誤ってマクロキーを押してしまうと予期せぬ動作を引き起こしてしまうことがあります．

3行目の「mp[,」は，システム変数のmp(ページングモード)を退避することを意味します．このマクロは文字列の検索を行うため，そのままではページングモードが切り換わってしまいます．マクロは，それを使う前後でシステムの状態が変更されないように気をくばるべきです．「[」は一種の演算子

で，直前の値を退避します（アセンブラのpushに相当）．退避した値は，「]」演算子で変数に復帰させます（アセンブラのpopに相当）．値を退避する方法は先入れ後出し(スタック)形式ですから，値を復帰するときは退避した順とは逆に行います．これらの演算子は，システムの状態を変更させないようシステム変数の内容を退避／復帰させるために役立ちます．[ ]で対にして覚えておくとよいでしょう．

6行目の「:A」は，ラベルをあらわします．繰り返しや条件分岐の際には，こうしたラベルが使われます．>Aとすれば，このラベルにジャンプすることができます．

8行目の「(r)?{mp],.}」は，直前の検索命令#cが成功したかどうかを判定しています．失敗することのあるいくつかのコマンドは，システム変数のrを判定することで処理が成功したかどうかを確認できます．たとえば，カーソル移動(#e，#s，#d，#x)はそれ以上移動できない場合に処理に失敗したものとみなし，rに-1が代入されます．ページングモードが検索モード(S)のときは，#c（前方検索）や#r（後方検索）で文字列が見つからなければ処理が失敗したものとみなされrが-1になります．(r)は，マクロを終了させる条件としてもよく使います．

ここまで理解できたら，[CTRL]+[K]，と入力してマクロを再カスタマイズし，リスト1をオープンして[CTRL]+[O]を押してみてください．ファイル全体にマクロが機能したことがわかるでしょう．

今度は，リスト5から関数名のプロトタイプ宣言だけを取り出すことを考えます．

「関数のプロトタイプ宣言」をどのように区別して取り出せばよいでしょうか．もともとC言語は，めんどうな構文解析を行って関数定義などを調べているわけですから，たかだかマクロでそれと同じレベルのことは要求できません．したがって，問題はいかに簡単化してマクロに適用させやすくするか，ということになります．

関数定義の特徴を見てみましょう．定義部には，関数引数を囲むための1対のカッコ()があります．しかし，これだけでは関数呼び出しやifやwhileの条件式と区別できません．行頭が英字であるというのは条件としてよさそうですが，データの定義文と区別がつかない可能性がありますし，VZの機能では英字を検索するのはちょっとややこしくなりそうです．そこで，次の行にある「{」を利用することを考えます．「{」だけでは，ブロックの始まりを示すところすべてが検索されてしまいますが，行頭に「{」があるのは，たいてい関数本体を記述しはじめるときだけでしょう．先頭の文字を見つけるときは，検索文字列として

LIST リスト5 sample.c 5

```
/*
 *    文字列の連結
 */
char far * far _fstrcat(char far *dest, const char far *src)
{
    char far *p, far *q;

    p = dest;
    q = src;
    while (*p)
        ++p;
    do {
        *p++ = *q;
    } while (*q++);
    return dest;
}

/*
 *    文字の検索
 */
char far * far _fstrchr(const char far *str, int c)
{
    while (*str) {
        if (*str == (char)c) return str;
        ++str;
    }
    return NULL;
}
```

LIST リスト6 プロトタイプ宣言を取り出すマクロ 6

```
* MACRO

81 ^O "プロトタイプ宣言の取り出し"
    ?.                  ; 編集モードでのみ有効
    #^ #<               ; ファイルの先頭へ移動
    mp[,                ; ページングモードの退避
    #F "Yn{" #m         ; 検索文字列の設定
:A                      ; 繰返しラベル
    #c                  ; 行頭の '{' を検索
    (r)?{mp], .}        ; 検索に失敗したら終了
    #e                  ; 1行前に移動
    #k                  ; その行をブロックバッファにコピー
    #x                  ; 1行下に移動
    >A                  ; マクロの繰返し

*
```

「¥n＋文字」を指定することで見つけられます．¥nは改行文字つまり行末ですが，その次は必ず行頭だからです．先頭の「{」が見つかったら，その上の行をブロックバッファにコピーします（キー操作では[SHIFT]＋[F8]に相当）．ここで再度検索すると，いま見つけた文字列に一致してしまいますから，いったんカーソルを下に移動して次の関数を検索します．

この手順をマクロにするとリスト6のようになります．

このマクロでは，見つけたプロトタイプ宣言の内容をすべてブロックバッファに退避します．したがって，テキスト自身は何も変更されません．ブロックバッファの内容は，F9（#i，行・ブロックの挿入）で順次テキストに挿入していくことができます．

ちょっと違うマクロを作ってみましょう．Cで条件コンパイル（#if〜#endif）を多用していると，これらの対応を調べたいことがあります．しかし，条件コンパイルのために使われる指令は#if，#ifdef，#ifndef，#else，#elif，#endifという複数の指令が存在します．したがって複数の文字列を同時に検索しなければならず，少々ややこしくなります．テキストを1文字ずつ調べるマクロを記述することもできるのですが，VZのマクロはそれほど高速には処理されません．本体でサポートされている機能を使うほうがはるかに効率がよいですし，そうするためには処理内容をどのように簡略化できるか考えなければなりません．

今回の条件では，調べるべき文字列を#if〜と#e〜という2種類の形式にわけることができます．この形式で他のコンパイル指令をあらわす文字列はありません．#if〜という形式は必ず条件コンパイルの先頭になり，条件コンパイルが終了するのは#endifのときだけです．#endifは，実際には#enまで調べれば十分です．もちろん，記述にまちがいがあったら正しい判定はできなくなりますが，それはコンパイラの仕事でエディタの仕事ではないと思って諦めましょう．条件コンパイルはネストまで考慮しなければいけないので少々めんどうですが，リスト7のようなマクロで実現できます．

LIST 7　リスト7　条件コンパイルの対応を検索

```
* M マクロ

82 ^KO "条件コンパイルの対応位置を検索"
        ?.
        mp[,            ; ページングモードの保存
        &s              ; 画面更新の一時停止（サイレントモード）
        n-,             ; ネストレベルの初期化
        #F "¥n#" #m     ; 検索文字列の設定（位置マーク）
        #<              ; 行頭へ移動し
        (cd!='#')?>Q    ; 先頭文字が#かどうか調べる
        #d              ; 次の文字
        (cd=='e')?>E    ; eかどうか調べる
        (cd!='i')?>Q    ; iかどうか調べる
        #d              ; 次の文字
        (cd!='f')?>Q    ; （iの次が）fかどうか調べる
:F                      ; #if〜、#elif、#elseの処理（前方検索）
        #c (r)?>Q       ; 前方検索
        f+,
        &10             ; 行の指令の意味を解析
        (n<0)?>X        ; ネストレベルが0で、#e〜であれば停止
        >F              ; 繰返し
:E
        #d              ; （eの）次の文字が
        (cd=='l')?>F    ; lかどうか調べる
:B                      ; #endifの処理（後方検索）
        #< #r (r)?>Q    ; カーソルを行頭へ移動し、後方検索
        f-,
        &10             ; 行の指令の意味を解析
        (n>0)?>X        ; ネストレベルが0で、#i〜であれば停止
        >B              ; 繰返し

:Q      #U >Z           ; 対応する位置がみつからなかったので、元の位置へ戻る
:X      #<              ; 行頭へ移動
:Z      &d              ; 画面更新を再開（通常モード）
        mp],            ; ページングモードを復帰

10:     #d              ; （#の）次の文字
        (cd=='i')?>I    ; iかどうか調べる
        (cd!='e')?.     ; （iでなく）eでもなければ終了
                        ; "else", "elif", "endif" ?
        #d              ; 次の文字が
        (cd=='n')?{n--, .} ; nであれば、#endif
        (n==0&&f)?n--,
        .
:I                      ; "if" ?
        #d              ; （#の）次の文字
        (cd=='f')?n++,  ; （iの）次の文字がfかどうか調べる
        .

*
```

少し長いのですが，新たなマクロの要素をいくつか含んでいるのでじっくりと観察してください．ここでは，nという変数を使って条件コンパイルのネストレベルを保持しています．nは，#if〜に出会うたびに1ずつ加算され，#endifに出会うたびに1ずつ減算されます．ただし，前方参照においては#endifだけでなく#elseや#elifでも停止させたいので，前方参照でもっとも外側にいるときは（つまりネストレベルが0のとき）は#elseや#elifでもnが減算されます．前方参照のときは，このnが負になれば条件コンパイルの外側に出た，つまり対応するカッコが見つかったことがわかります．逆に，後方参照のときは，nが正になったかどうかで判断できます．

マクロの先頭のほうにある&sとは実際のテキスト画面を更新しないサイ

おもなUNIXコマンド
rmdir　ディレクトリの削除

レントモードをあらわします．画面を更新する必要がないので，処理の高速化がはかれるほか，マクロ処理のチラつきをなくすことができます．マクロを終了するときは，&dを使って正しく実行結果を画面に表示するようにしなければなりません．

このマクロでは，cdという変数が条件判定のために多用されています．cdは，カーソル位置にある文字の文字コードをあらわします．残念ながら，VZのマクロにはカーソル位置にある文字列を比較する機能がありません．文字列を比較するときは，カーソル位置を#d（右移動）で移動させながら1文字ずつ比較していかなければならないのです．このようにして比較した結果が行頭からの文字列が#ifであるか#elであれば前方検索し，#enであれば，後方検索します．

サブマクロの呼び出し

マクロの中で&10という表記は，マクロ中に記述されたサブマクロを呼び出すことをあらわします．プログラムのきまった処理をサブルーチンにまかせて効率化をはかるのと同じように，VZのマクロでもサブマクロを定義してメインのマクロからそれを呼び出すことができます．サブマクロは，番号+:（コロン）に続いて定義します．ちなみに，マクロの中にサブマクロを定義しなければ他のマクロを呼び出すことになります．ここでは，カーソル行が#if〜であるか#e〜であるかを調べて，ネストレベルの更新を行っています．

実際には，C言語では（ANSIに準拠している限り）行頭と#，#とifの間にはスペースを入れることができます．このマクロでは，こうしたプログラムに対応できません．いまのVZの検索機能のまま行頭のスペースまで判定していたら，かなりの時間が費やされてしまうことになるでしょう．しかし，VZがワイルドカードや正規表現での検索をサポートしてくれれば簡単に対応できるのも事実です．将来のバージョンに期待することにしましょう．

インスタントマクロ

VZのマクロには，もう1つ便利な側面があります．それは，[ESC]，[@]によるインスタントマクロの実行です．

[ESC]，[@]と押すとマクロ文の入力が促されますので，たとえば「(70){'='}」と入力してみてください．()で式を評価した直後に{}ブロックを記述すると，そのブロックを評価した値の数だけ繰り返すことをあらわします．つまり，これで70個の「=」文字がテキストに挿入されるはずです．ドキュメントを書いていて，項目を区切るときなどに使えますね．また，「256*16」と入力すると，画面の最下位行に4096と表示されます（他の情報もいっしょに表示されます）．これは，インスタントマクロではマクロ式を評価した結果が画面の最下位行に表示されるので，このような簡易電卓として使うこともできるわけです．

マクロを再カスタマイズするためには，エディタでマクロテキストを作成しなければなりませんが，インスタントマクロならばそのようなめんどうなことはありません．この中では，たいていのマクロ文を正しく認識してくれるので，繰り返し文や条件判定を利用することもできます．文字列については，VZの構造上扱うことができません．

インスタントマクロは，マクロを書く場合に機能をテストしてみることも簡単にできますから，ぜひ活用してください．

VZのマクロを使いこなすこととVZをエディタとして使いこなすことは似ています．VZのエディタとしての機能をよく把握すれば，効率のよいマクロを組むことがやさしくなります．カット&ペーストのためのブロックバッファが先入れ後出し（スタック）の形式になっていることがわかれば，テキストの中にばらまかれている情報をまとめるマクロも組みやすいでしょう．#G（[CTRL]+[L]，文字列の取得）と#I（[CTRL]+[Q]，[I]，文字列の複写）を組み合わせれば，ブロックバッファを変更せずに文字列をコピーできます．

VZもバージョン1.5が出荷されてから2年が過ぎました．マイナーな変更は行われたものの，大幅な機能追加などは行われていません．執筆時点ではバージョンアップに関する具体的な動きもチラホラ聞こえてきていますが，この号が発売される頃にはどのようになっているでしょうか．マクロの領域が圧迫されているという話もよく聞きますので，そうした制限ができるだけなくなるようになれば，マクロ屋として嬉しい限りです．

VZエディタ

●**製品名**：VZ EDITOR V.1.5
●**価格**：9,800円（PC-9800シリーズ，J-3100シリーズ，AX，IBM PC用）
5,000円（DOS/V用）
●**問い合わせ先**：ビレッジセンター
TEL 03-3239-2187

マクロの達人 Brief

強力なツールをC構文で書ける

梓　史郎

ツールとしてのエディタ

プログラミングツールというと，まず頭に浮かぶのはコンパイラやデバッガで，エディタをその仲間に入れる人は少ないでしょう．そのわりには，話がエディタとなると自分の主張に固執する人が多いのも事実です．『Cマガジン』の91年11月号には「Text editors」と題した1文が掲載されていましたが，お読みになったでしょうか？　「プログラマにエディタについて語らせたら，話は尽きないだろう．ほかのことでは協調的な人でも，自分の好みのエディタ以外によい点をつけた人を軽蔑する……私はテキストエディタに関してこうした現象にずっと懸念を抱いてきた．……」とP.J.Plauger氏は語っています*1（この話はたいへん示唆に富んでいて面白いのでぜひご一読をお勧めします）．

プログラマが自分の使っているエディタに執着するのは，ある意味では，日ごろ使いなれた機能以外には無関心で，無意識にエディタの使い方に一定の枠をはめてしまっているからだともいえるのではないでしょうか．というわけで，本稿ではプロフェショナルやそれに準じるヘビープログラマ，あるいは新しい物には目がない，という方に「エディタをツールとして活用しよう」という，日常より少し高い目線からエディタを見なおすことを提案したいと思います．

ここで取りあげるエディタはBriefです．最近では，その名を少しは耳にするようになってきましたが，まだ1度も使ったことのないという方もあることでしょう．Briefはプログラミング用のエディタと称しているだけあって，ほかのエディタに見られないプログラミング用のさまざまな機能があります．また，強力なマクロ機能も備わっています．ここでは，Briefのプログラミングエディタとしての機能と，そのマクロ機能の一端を紹介します．

プログラミングエディタとしての機能

Briefには，プログラミング用エディタとしてのさまざまな機能があります．プログラミング用としての機能とは何か――Briefで実現されている機能は，プログラミングをする上でエディタにはどんな機能が必要か，という問に対するひとつの解答でもあります．

ソースコードの編集

一般の文書は禁則等を除くと形式は自由で，これといった規則性はありません．ところがプログラミング言語は，それぞれ独自の構文をもっています．また，プログラムの編集ではインデントを多用し，それが重要な意味をもちます．プログラミング用のエディタでは，そうしたプログラミング言語独自

の規則や習慣に合った入力や編集が効率よくできることが望ましいわけです．Briefにはインデント機能，7種類のプログラミング言語で使用できるテンプレート機能があります．

・自動インデント（Regular，普通のインデント）

行の最後でリターンキーを押すと，次の行が自動的に前行と同じ開始位置からはじまります．

・スマートインデント（Smart）

プログラム言語の文脈に対応したインデントを自動的に行います．文の終わりなど文の一部を認識し，インテリジェントにインデントの位置を判断します．C言語の場合を例に見てみましょう（図1）．行末でリターンキーを押すとカーソルは自動的に次の行の▼へ移動します．行末の‘;’や‘{’の有無によって次の行の開始位置を調整するという，プログラミング用のエディタならではの自動インデント機能です．

・テンプレート（Template）

プログラムの構文上で，キーワード部分は固定的です．テンプレートは，キーワードとそれに関連する構文の骨格を自動的に生成します．たとえばdo-while構文の場合，dとスペースを入力すると，図2に示すような構文の骨組みを作り出します．▼は文を作り出した後，カーソルが移動する位置です．

また，if文の最後で‘{’を入れてからリターンを押すと，‘}’を自動的に生成し，カーソルを次の文の入力位置に動かすことも行います（図3）．

C言語の場合，テンプレート機能が生成する構文は12種類あります．また，好みのコーディングスタイルに合わせてインデント位置を変えるオプションもあります．このテンプレート機能は次の7言語に適用できます．

Ada
BASIC
C
COBOL
FORTRAN
Modula-2
Pascal

FIG 1　図1　スマートインデントの例

```
total=sub1+sub2↓
   ▼
   sub3+sub4;↓
▼
または

 if (act==go)↓
 {↓
    ▼
```

FIG 2　図2　do-while構文のテンプレート

```
do-while構文のテンプレート
     do
         {
         ▼
         }
     while( );
```

FIG 3　図3　if文のテンプレート

```
if(gap > LIMIT) {↓
   ▼
}
```

・インデントの変更

コーディングの最中には，複数の行をまとめて左右にスライドさせ，インデントの深さを変えたいという場合が出てきます．if文にさらに条件を追加したため，それ以下の実行文のインデントを1段深くするといった場合です．Briefでは，移動させたい範囲を反転させておけば，［TAB］で右へ，［SHIFT］+［TAB］で左に移動させることができます（図4）．

矩形の領域でも同じように左右に移動させることができます．矩形の水平移動は，定数表の編集などの際に便利です．

FIG 4　図4　複数の行をまとめて左右に移動する

```
─────────────── original.c ───────────────
 case 'C':
   if (context >= 0)
     fatal ("context length specified twice");
   {
     char *p;
     for (p = optarg; *p; p++)
       if (*p < '0' || *p > '9')
    fatal ("invalid context length argument (-C option)");
   }
   context = atoi (optarg);
─────────────── shift.c ───────────────
 case 'C':
   if (context >= 0)
     fatal ("context length specified twice");
   {
         char *p;
         for (p = optarg; *p; p++)
           if (*p < '0' || *p > '9')
         fatal ("invalid context length argument (-C option)");
   }
   context = atoi (optarg);

                         行: 163  カラム: 1   ↑  16.08
```

プログラミング言語別の対応

プログラミングといっても，プログラミングに使用する言語はたいてい1

種類ではありません．多くのソフトハウスではCとアセンブラ，大学や研究所でしたらPascalとFORTRANあるいはLispなどを使うことが多いのではないかと思います．

ところがそれぞれのプログラミング言語ごとに，最適なタブ位置やコンパイラの起動コマンド，インデントの方式などが異なっています．Briefではプログラミング言語ごとに編集環境が設定でき，しかも編集ファイルを切り換えると，自動的にそのファイルに対応した編集環境に切り替えてくれるのです．図5に編集対象をCからFortranに切り替えた場合の編集環境の変化を示します．そのファイルがどのプログラミング言語で書かれたものかは，ファイル名の拡張子で判断されます．拡張子別に表1で示す項目が設定できます．図6は，パラメータを拡張子に対して設定しているところです．

CHART 1　表1　拡張子ごとに設定できる項目

タブ停止位置	任意の位置にタブを設定
インデント方式	言語に合わせた自動的な字下げや構文の自動生成を指定
英文テキスト整形	英文書作成用の自動的なワードラップ，日本語対応のマージン設定と自動折り返し用のマクロがパッケージされている

ほかのツールへの対応

プログラムの開発ではコンパイラをはじめ多くの開発ツールを利用します．エディタでプログラムを作成してコンパイル，コンパイル・エラーがあれば再度エディタにもどってソースを編集する――このサイクルを繰り返しながら完成に向かうのです．コンパイル・エラーがなくなっても，テストの結果がおかしいときは，またエディタでソースの修正にもどります．つまり，プログラム開発のサイクル中で，エディタはつねに仕事の中心にいることになります．エディタを終了せずにほかの作業を行い，終了後はまた編集作業に復帰できれば，それだけスムーズに開発が進むことにもなります．

・コンパイル

Briefでは，エディタの中からそのままワンタッチでコンパイルできます．30種にものぼるコンパイラの必要メモリサイズ，コンパイル・コマンドのイメージなどが内蔵されているので，どのコンパイラを使うかを指定するだけで，Brief内からコンパイルができます．

コンパイルの開始キーを押すと，コンパイルを行い，コンパイル・エラーが見つかると自動的にエディタにもどって，カーソルが修正場所に誘導してくれます．ただちに修正できるので，極めてスピーディに作業が進みます．使用するコンパイラは，ファイル名の拡張子ごとに設定できます(図6)．

・バージョン管理ツール（PVCS）

Briefには，ソースコードのバージョン管理ソフトPVCS*2を連動して動かすためのインタフェースが用意されています．このPVCSインタフェースは，編集開始と最新版の取り出し，編集の終了と新版の登録を一体化しているので，見かけ上はエディタの操作だけですむようになっています．つまり，通常のBriefの操作を行うだけでPVCSが実行されるのです．

FIG 5　図5　CとFortranの編集環境

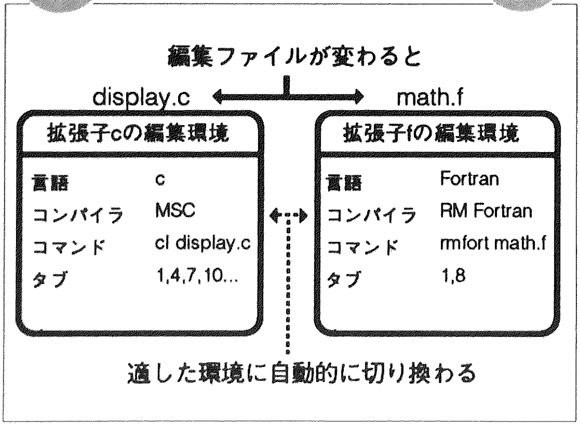
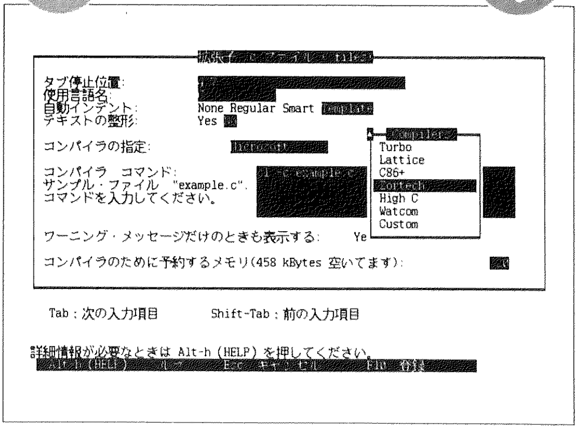


FIG 6　図6　拡張子ごとに環境を設定する



・データベース言語用プログラミング・ツール

Briefでは，dBASE III＋やdBASEIVあるいはFoxBASE+，FoxPro，ClipperおよびdbXL／Quicksilverなどのデータベース言語のプログラミングも行えます[*3]．これらの機能はマクロによって実現されており，前出のテンプレートのような構文ジェネレータのほかに，コメンタ，ソースライブラリ，特化したサーチ，プリプロッセッサ，印刷機能などdBASE系の言語のプログラミングに便利なさまざまな機能があります．

・DOSコマンドとの連携

DOS上で実行可能なプログラムならBriefの中から起動でき，マクロを使って実行の結果を受け取ることができます．これにより自作ツールやオンラインソフトとの連携がはかれます．

豊富なサーチ機能

プログラムでは定数，変数名，関数名など共通的なシンボルが複数のソース間で強い関連をもっています．そのため，プログラムの編集では，ほかのソースコードを調べたり，関係する複数のファイルを同時に編集したりする必要があります．

・タグ機能

関数は，プログラムの基本情報です．プログラムのロジックを調べるときは，関数の呼び出し関係を手がかりにします．逆に作成中は，パラメータの設定などのために，関数の定義を参照します．そこで，任意の場所から関数へ，その名前だけでアクセスするタグ機能がたいへん便利です．Briefでは，編集中の画面上で，関数名にカーソルを合わせ，タグ・コマンドを起動すると，ただちに該当の関数定義の内容が画面に表示されます．

・正規表現によるサーチ

正規表現はワイルドカードに似ていますが，はるかに高度なパターンを指定できます．プログラミング中は，さまざまなシンボルを扱う関係から，こ

LIST　リスト1　PAT_EDT.CB　1

```
/**********************************************************************/
/*                 p a t _ e d t   マクロ                              */
/*--------------------------------------------------------------------*/
/*                                                                    */
/*   目的：    複数のファイルを対象に特定のシンボルの使用箇所を調べる。 */
/*             あるいは、関数の定義場所を調べる。                      */
/*             見つかった場所から任意の場所を選び、調べたり、編集した  */
/*             りできる。また、これを繰り返す事ができる。              */
/*--------------------------------------------------------------------*/
/*                                                                    */
/*   使い方：                                                         */
/*       準備：                                                       */
/*         ・調べたいファイルは全て編集状態にしておく。Ｂｒｉｅｆは編 */
/*           集ファイル数に制限がないので都合がよい。                  */
/*       起動：                                                       */
/*         ・このマクロを起動する。Ｆ１０を押してから pat_edt と入力  */
/*           する。                                                    */
/*         ・関数名を調べるのか、シンボルを調べるのかをプロンプトに応 */
/*           えて選択する。y は関数、それ以外はシンボルを調べる。      */
/*         ・関数名でない場合は、プロンプトに応えてサーチのパターンを */
/*           指定する。                                                */
/*                                                                    */
/*       全てのファイルを対象にしてサーチが始る                        */
/*         ・見つかった場所の一覧が表示される。そこから調べる、あるい */
/*           は編集したい場所を選択する。                              */
/*                                                                    */
/* 一覧表からの選択：参照一覧が表示されている状態で以下を選択できる。 */
/*           ●リターンを押す                                          */
/*             ・選択した場所が表示され編集できる状態になる。          */
/*             ・調べたり、編集ができる。終ったらＥＳＣを押す。再び一 */
/*               覧に戻り、別の場所を選択できる。                      */
/*           ●ＥＳＣを押す                                            */
/*             ・マクロを終了し、起動前の状態に戻る。                  */
/*           ●ＣＴＲＬ＋ｇを押す                                      */
/*             ・選択した場所が編集状態になる。一覧表には戻らず終了する。*/
/*           ●ＣＴＲＬ＋ｆを押す                                      */
/*             ・カーソル行を上の画面に表示し、下の画面で他のファイル */
/*               を選択できる。下の画面では、選択を繰り返す事ができる。*/
/*               複数のファイルを同時に画面上で比較できる。            */
/*                                                                    */
/**********************************************************************/

/**********************************************************************/
/*                                                                    */
/*   謝辞：                                                           */
/*       本プログラムは以下の方々のアイデアを元に再コーディングしたもの */
/*     です。                                                          */
/*                                                                    */
/*   Special thanks for Mike Packard and forerunners.    Shiro Azusa  */
/*--------------------------------------------------------------------*/
/*                                                                    */
/*   This version modified by Mike Packard, San Diego, CA on 9/30/89  */
/*                                                                    */
/*   Original filename was NEWGREP.M                                  */
/*   Original source code modified by Zev Handler 5/19/89             */
/*   Obtained from the Solution Systems BBS on or about 8/89          */
/*                                                                    */
/**********************************************************************/

void
pat_edt()
{
    int             line_at, list_buffer, num_buffer, cur_buffer, cursr_line,
                    cursr_col, window_line, window_col, old_buffer,
                    re, list_key, mul_win;

    string          buf, pattern, line_no, line;
    global          old_buffer, list_buffer, cursr_line, cursr_col, list_key,
                    mul_win;

    mul_win = 0;
                                        /* 繰り返し編集中のキー */
    keyboard_push();
    assign_to_key("<PgUp>",    "pgup_litebar");
    assign_to_key("<PgDn>",    "pgdn_litebar");
    assign_to_key("<Esc>",     "_list_exit");
    assign_to_key("<Enter>",   "edit_global");
    assign_to_key("<Up>",      "up_litebar");
    assign_to_key("<Down>",    "down_litebar");
    assign_to_key("<Ctrl-g>",  "global_goto");
    assign_to_key("<Ctrl-f>",  "fix_one");
    assign_to_key("<Del>",     "global_del");
    list_key = inq_keyboard();
    keyboard_pop(1);
                                        /* カーソル位置を保存   */
    inq_position(cursr_line, cursr_col);
    inq_top_left(window_line, window_col);

                                        /* 編集中のバッファーを調べる */
    old_buffer = inq_buffer();
    do {
        ++num_buffer;
        cur_buffer = next_buffer();
        set_buffer(cur_buffer);
        inq_names(NULL, NULL, buf);
        if (list_buffer && inq_modified(cur_buffer)){
            delete_buffer(list_buffer);
            list_buffer = 0;
        }
    } while (old_buffer != cur_buffer);

    if (list_buffer){                   /* 前回調べた結果が残っている */
        if (!get_parm(0, buf, "再度調べ直しますか [y/n]:", 1, "n"))
            return;
        if (index("yY", buf)){
            delete_buffer(list_buffer);
            list_buffer = 0;            /* それを使わない       */
        }
    }

    if (list_buffer == 0){              /* 結果は無い　新たにバッファーを作る */
        list_buffer = create_buffer("一覧", "$global$.$$$", 1);
        if (inq_marked()) {
            int             num_chars;

            num_chars = inq_mark_size();    /* マーク部分をパターンにする */
```

```
            swap_anchor();
            pattern = read(num_chars);
            raise_anchor();
            re = 0;
        } else {
            re = 1;
            if (!get_parm(0, pattern, "関数を探しますか[y/n]: ", 1))
               return;
            if (index("Yy", pattern))        /* 関数サーチ      */
               pattern = "<[^ ¥t#/¥¥*;¥n¥¥{¥¥}(^L]*)[^;]";
                                              /* パターンサーチ */
            else if (!get_parm(0, pattern, "サーチパターン(RE): ")) return;
        }
        if (!strlen(pattern)) {
            error("パターンが指定されていません");
            return;
        }
                                              /*  全バファーをサーチする */
        while (num_buffer > 0) {
            inq_names(NULL, NULL, buf);
            --num_buffer;
            message("サーチ中: %s  残: %d ﾌｧｲﾙ", buf, num_buffer);
            cur_buffer = inq_buffer();

            top_of_buffer();
            while (search_fwd(pattern, re)) {
               beginning_of_line();
               line = read();
               inq_position(line_at);
               set_buffer(list_buffer);
               beginning_of_line();
               insert(buf);
               move_abs(0, 14);
               sprintf(line_no, "%5d  ", line_at);
               insert(line_no);
               move_abs(0, 20);
               insert(line);                  /* サーチで見つかった行を  */
               set_buffer(old_buffer);        /* 一覧バッファーに入れる  */
               move_abs(++line_at, 1);
            }
            set_buffer(cur_buffer);
            old_buffer = next_buffer();
            set_buffer(old_buffer);
                                    /* サーチ中に何かキーが押されて    */
                                    /* いたら中断する                  */
            if (inq_kbd_char()) {
               beep();
               keyboard_flush();
               num_buffer = -1;
               set_buffer(old_buffer);
               attach_buffer(old_buffer);
               error("Search aborted.");
            }
        }   /* end while */
    }

    if (num_buffer != -1) _show_list();     /* 一覧表を表示する      */
    if (mul_win)delete_edge(0);             /* マルチウインドウに    */
                                            /* なっていたら閉じる    */
    set_top_left(window_line, window_col);  /* 画面を戻す            */
    goto_old_line(cursr_line);
    move_abs(NULL, cursr_col);
}

_list_exit()
{
    message("");
    exit();
}


_show_list()
{
    set_buffer(list_buffer);                  /* 一覧表を表示する     */
    attach_buffer(list_buffer);
    beginning_of_line();
    top_of_buffer();
    drop_anchor(3);
    refresh();

    keyboard_push(list_key);                  /* キーを一覧表用にする */
    message("RET=編集, CTRL+f=固定, CTRL+g=移動, ESC=終了");

    process();
    keyboard_pop();                           /* キーを元に戻す       */
    set_buffer(old_buffer);                   /* 元のバッファーにする */
    attach_buffer(old_buffer);
}


int   pic_list()
{
    int              buflines;
    string           selbuf, curbuf;

    beginning_of_line();
    selbuf = trim(read(12));                  /* 一覧表から選択内容を */
    move_abs(0, 14);                          /* 取り出す             */
    buflines = atoi(read(5), 1);              /* 行番号               */
    inq_names(NULL, NULL, curbuf);
    while (selbuf != curbuf){                 /* ファイルをロードする */
      set_buffer(next_buffer());
      inq_names(NULL, NULL, curbuf);
    }
    attach_buffer(inq_buffer());
    return(buflines);
}

edit_global()                                 /*  RET で起動          */
{
    int              buflines;

    save_position();
    buflines = pic_list();                         /* 選択されたファイルを編  */
                                                   /* 集する                  */
    goto_old_line(buflines);                       /* パターンの行へ          */
    refresh();

    keyboard_pop(1);
    copy_keyboard(inq_keyboard());                 /* 編集時のキーにする      */
                                         /* inhibit execution */
    assign_to_key(inq_assignment("edit_file",1),          "nothing");
    assign_to_key(inq_assignment("edit_prev_buffer", 1), "nothing");
    assign_to_key(inq_assignment("edit_next_buffer", 1), "nothing");
    assign_to_key(inq_assignment("output_file", 1),       "nothing");
    assign_to_key(inq_assignment("altx", 1),              "nothing");
    assign_to_key(inq_assignment("write_and_exit", 1),    "nothing");
    assign_to_key(inq_assignment("delete_curr_buffer", 1), "nothing");
                                        /* switch ESC  to exit */
    assign_to_key("<Esc>", "_edt_exit");
    message("編集できます. ESC=一覧");

    process();
    keyboard_pop();                                /* 一覧表に戻す         */
    set_buffer(list_buffer);
    attach_buffer(list_buffer);
    keyboard_push(list_key);
    restore_position();
    drop_anchor(3);
    refresh();
}

_edt_exit()
{
    message("RET=編集, CTRL+f=固定, CTRL+g=移動, ESC=終了");
    exit();
}

fix_one()                                        /* CTRL+f で起動        */
{
    if (mul_win){
       beep();
       return;
    }
    goto_old_line(pic_list());                   /* 選択行に移る         */

    create_edge(2);                              /* 下にウィンドを作る   */
    set_buffer(list_buffer);                     /* そこへ一覧を表示     */
    attach_buffer(list_buffer);
    beginning_of_line();
    mul_win = 1;
    message("RET=編集, CTRL+f=固定, CTRL+g=移動, ESC=終了");
}

global_goto()                                   /* CTRL+g で起動         */
{
    cursr_line = pic_list();                    /* 選択ファイルを編集し  */
    old_buffer = inq_buffer();                  /* 指定行へ飛ぶ          */
    cursr_col = 1;
    message("通常の編集です");
    exit();
}

/*    一覧表のカーソル移動     */

up_litebar()                                     /* 1行上へ             */
{
    raise_anchor();
    up();
    drop_anchor(3);
}

down_litebar()                                   /* 1行下へ             */
{
    raise_anchor();
    move_rel(2);
    if (read(1) == "¥n")up();
    up();
    drop_anchor(3);
}
                                                 /* 一覧表から削除      */
global_del()
{
    raise_anchor();
    delete_line();
    drop_anchor(3);
}

                                                 /* スクロールアップ    */
pgup_litebar()
{
    raise_anchor();
    page_up();
    drop_anchor(3);
}
                                                 /* スクロールダウン    */
pgdn_litebar()
{
    raise_anchor();
    page_down();
    search_back("<[^ ¥n]");
    drop_anchor(3);
}
```

FIG 7　図7　参照箇所の一覧

```
                                 一覧
io.c           135   compareseq (xoff, b, yoff, b - d);
analyze.c      233 compareseq (xoff, xlim, yoff, ylim)
analyze.c      279    compareseq (xoff, b, yoff, b - d);
analyze.c      284    compareseq (b, xlim, b - d, ylim);
analyze.c      449    'compareseq' always chooses the one at the beginning,
analyze.c      723   compareseq (0, filevec[0].nondiscarded_lines,
analyze.c      795   compareseq (xoff, b, yoff, b - d);
context.c      208   compareseq (xoff, b, yoff, b - d);
context.c      241   compareseq (xoff, b, yoff, b - d);
context.c      280   compareseq (xoff, b, yoff, b - d);
diff.c          54   compareseq (xoff, b, yoff, b - d);
dir.c           53   compareseq (xoff, b, yoff, b - d);
dir.c           79   compareseq (xoff, b, yoff, b - d);
dir.c          175   compareseq (xoff, b, yoff, b - d);
ed.c            82    compareseq (xoff, b, yoff, b - d);

RET=編集, CTRL+f=固定, CTRL+g=移動, ESC=終了        行: 3   カラム: 1   ↑  17:28
```

FIG 8　図8　3番目の参照箇所

```
                              analyze.c
      compareseq (xoff, b, yoff, b - d);
      /* This used to use f instead of b,
         but that is incorrect!
         It is not necessarily the case that diagonal d
         has a snake from b to f.  */
      compareseq (b, xlim, b - d, ylim);
    }
     }
}
♀
/* Discard lines from one file that have no matches in the other file.

   A line which is discarded will not be considered by the actual
   comparison algorithm; it will be as if that line were not in the file.
   The file's `realindexes' table maps virtual line numbers
   (which don't count the discarded lines) into real line numbers;
   this is how the actual comparison algorithm produces results
   that are comprehensible when the discarded lines are counted.

   When we discard a line, we also mark it as a deletion or insertion
   so that it will be printed in the output.  */
編集できます. ESC=一覧                           行: 279  カラム: 1   ↑  17 29
```

のサーチ機能を活用するのがコツです．また，マクロの中で使うと絶大な威力を発揮します．

アンドゥと自動保存

編集した結果を，それより前の状態にもどすアンドウは，プログラマの強力な味方です．実際，Briefの300ステップも前の状態に復元できるアンドウの味を覚えると，そのとりこになります．私はこのアンドウ機能だけをとってみても，Briefはプログラマにお薦めできるエディタだと思います．

さらに，万一の場合の備えとして自動的に取られるバックアップや編集途中の最新イメージの自動保存など，何かハプニングがあっても，振り出しにもどらないための備えは，絶大な安心感を与えてくれます．

Cライクなマクロ機能

エディタは，プログラマにとって筆記用具と消しゴム，糊とはさみに相当する身近な道具ですから，使いやすく，手になじむ物でなくてはなりません．マクロで拡張やカスタマイズができると，自分にとって最適な環境へ容易に変身させることができます．とはいっても，一般にマクロは独自の解りにくい文法を採用しているので，つい尻込みしてしまいます．

BriefのマクロはC構文で書けてスクリーンデバッガでデバッグできるので，プロやパワーユーザには最適です．エネルギーは，創造的なプログラミング環境を作るために使いましょう．最後にマクロで作った開発ツールを示します(リスト１)．実用的で強力なツールがマクロで実現できることがおわかりいただけると思います．

複数ファイル間で検索ジャンプするマクロ

リスト１のマクロは，サーチ機能を生かしてプログラマの作業をサポートするツールです．プログラミングでは，どこか１か所を変更すると関係する複数のソースをなおさなくてはならない，これは常識です．このような場合には，関連する変数名，関数あるいは定数などの使用箇所を複数のソースに渡って調べる必要があります．このマクロは漏れなくすべての参照箇所を調べたり，変更したりするためのものです．

このマクロを起動すると，現在編集中の複数のソースを対象に特定のシンボルが使われているすべての場所を探し出して一覧表を表示します．一覧表からは任意の場所を選択してジャンプすることができます．編集が終ったら，もう一度一覧表にもどり，ほかの場所へのジャンプ，編集を繰り返すことができます．

では，実際にこのマクロを使ってみましょう．ここでは関数compareseqに関連した場所を修正します．マクロを実行する前に，調べたいファイルはあらかじめすべて編集状態にしておきます．

①このマクロを起動する．[F10]を押してからpat_edtと入力する．
②関数名を調べるのか，シンボルを調べるのかをプロンプトに応えて選択する．compareseqは関数なので[y]を入力する．
③パターンにcompareseqを指定する．

すべてのファイル中でcompareseqを使っている場所が一覧表示されます（図７）．

④３番目を選択する（図８）．

ここでは普通の編集操作ができます．

FIG 図9　再び一覧にもどる　9

```
                              一覧
io.c           135   compareseq (xoff, b, yoff, b - d);
analyze.c      233 compareseq (xoff, xlim, yoff, ylim)
analyze.c      279    compareseq (xoff, b, yoff, b - d);
analyze.c      284    compareseq (b, xlim, b - d, ylim);
analyze.c      449    `compareseq' always chooses the one at the beginning,
analyze.c      723   compareseq (0, filevec[0].nondiscarded_lines,
analyze.c      795   compareseq (xoff, b, yoff, b - d);
context.c      208   compareseq (xoff, b, yoff, b - d);
context.c      241   compareseq (xoff, b, yoff, b - d);
context.c      280   compareseq (xoff, b, yoff, b - d);
diff.c          54   compareseq (xoff, b, yoff, b - d);
dir.c           53   compareseq (xoff, b, yoff, b - d);
dir.c           79   compareseq (xoff, b, yoff, b - d);
dir.c          175   compareseq (xoff, b, yoff, b - d);
ed.c            82    compareseq (xoff, b, yoff, b - d);

RET=編集, CTRL+f=固定, CTRL+g=移動, ESC=終了        行: 12  ｶﾗﾑ: 1   ↑  17 35
```

FIG 図10　別のファイルに移る　10

```
                              dir.c
  compareseq (xoff, b, yoff, b - d);
  if (nonex)
    {
      dirdata.length = 0;
      dirdata.files = 0;
      return dirdata;
    }

  /* Open the directory and check for errors.  */
  reading = opendir (dirname);
  if (!reading)
    {
      perror_with_name (dirname);
      dirdata.length = -1;
      return dirdata;
    }

  /* Initialize the table of filenames.  */

  nfiles = 100;
  files = (char **) xmalloc (nfiles * sizeof (char *));
編集できます. ESC=一覧                                行: 53  ｶﾗﾑ: 1   ↑  17 35
```

FIG 図11　定義と参照箇所を同時に調べる　11

```
                              analyze.c
compareseq (xoff, xlim, yoff, ylim)
     int xoff, xlim, yoff, ylim;
{
  /* Slide down the bottom initial diagonal. */
  while (xoff < xlim && yoff < ylim && xvec[xoff] == yvec[yoff])
    ++xoff, ++yoff;
  /* Slide up the top initial diagonal. */
  while (xlim > xoff && ylim > yoff && xvec[xlim - 1] == yvec[ylim - 1])
    --xlim, --ylim;
                              dir.c
  compareseq (xoff, b, yoff, b - d);
  /* Loop while files remain in one or both dirs.  */
  while (i1 < data1.length || i2 < data2.length)
    {
      int nameorder;

      /* Compare next name in dir 1 with next name in dir 2.
    At the end of a dir,
    pretend the "next name" in that dir is very large.  */
編集できます. ESC=一覧                    行: 175  ｶﾗﾑ: 1   ↑  17:42

context.c
);
->link;
ast1, deletes, inserts;
      /* Turn this change into a hunk: detach it from the others.  */
      script->link = 0;
編集できます. ESC=一覧                    行: 241  ｶﾗﾑ: 1   ↑  17 41

analyze.c
- d);
 of b,
case that diagonal d
*/
lim);
  }
}
編集できます. ESC=一覧                    行: 279  ｶﾗﾑ: 1   ↑  17 39
```

下の参照画面が次々と切り替わる

修正が終ったら，ESCを押して再び一覧にもどります（図9）．

⑤12番目に移る（図10）．

今度は図10のように別のファイルに移ります．このように参照箇所の一覧にそって変更して行けば，安全で一貫性のある変更ができます．

複数のソースを比べる時のためにマルチウィンドウを利用して，2か所を同時に比較する機能も取り入れました．上の画面には関数の定義を表示し，下の画面にその参照箇所を次々と表示するといったことができます．一覧表の上で[CTRL]＋[f]を押すと，該当のソースが上画面に表示され，下画面に一覧表が表示されます．一覧表では，任意の場所を選択できます(図11)．編集してソースコードのライン番号が変わっても，一覧表はそのまま使えます．

欧米ではエディタはプログラミング用に，ワープロは文書の作成にとそれぞれの長所を生かして使い分けるのが普通のようです．重いワープロの代用として，軽くて速いエディタを使うのは日本ならではの事情ですが，プログラミングにもワープロ代わりのエディタ1本槍，というのはちょっと芸がないようにも思えます．用途に合った道具を使いわける．これが昔からの人間の知恵なのではないでしょうか？

＊1 「Programming on Purpose」 Computer Language May.1991 C Magazine Nov.Vol3.No11 野口修男訳 福富寛監修

＊2 PVCSはPOLYTRON Corp.の製品．日本では(株)ライフボートから発売されている

＊3 Global Tchnologies社製でSolution Systems社が販売している．

Brief

●製品名：Brief日本語版V.3.0
●価格：48,000円（東芝，NEC，AX協議会，IBM，富士通，DOS/Vの6機種対応パッケージ）
●問い合わせ先：日本システム(株) TEL 0425-23-4469

特集
プログラミングツール大集合

創刊１周年 特別企画［アンケート］

本誌に登場した37人に聞きました．あなたの使っているツールを教えて！

「プログラミングの役に立つツールには何があるのか？」その答えは，プログラムの目的や用途，実行の環境によって少しずつ違ってくるのではないだろうか．編集部では，この１年間に本誌で執筆していただいた著者，取材にご協力いただいたソフトウェア会社のプログラマの方々を対象に，プログラミングツールに関していくつかの質問を試みた．本誌で連載を担当している執筆者やソフトウェアの開発を職業とするプロのプログラマの生の声は，きっと読者プログラマ諸氏の参考になることと思う．

なお，年末をひかえて多忙を極める中，快く質問に答えてくださった方々にはこの場を借りて心からお礼申し上げます．

【質問の内容】

①　どんな種類のプログラムを作成することが多いですか？

②　よく利用するOSは何ですか．

③　開発に使用しているメインのコンピュータは何ですか．

④　よく使うプログラミング言語は何ですか．

⑤　それぞれの言語で，使用しているコンパイラは何ですか．

⑥　ふだんのプログラミングでコンパイラ製品の「統合環境」を利用しますか．する／しないの別とその理由もお聞かせください．

⑦　プログラミングに使用するエディタは何ですか．

⑧　使用しているデバッガは何ですか．

⑨　make，linkerはコンパイラ製品付属のものを使っていますか．

イ）使っている．

ロ）使っていない．代わりに自作プログラムを使用．

ハ）使っていない．ほかの製品を使用している．

ニ）使っていない．シェアウェア／フリーウェアを使用している．

⑩　ふだん利用しているその他のツール類（バージョン管理ツール，ドキュメント生成ツール，ステップ数カウントツール，画面作成ライブラリなど）があれば教えてください．

⑪　プログラマにお薦めのツールがあればぜひご紹介ください．

⑫　プログラミングに有用なツールとは，どんなものだと思いますか？　ご意見をお聞かせください．

⑬　本誌の読者に何かひとことお願いします．

秋山 登

91年４月号「Cプログラミングのスタイルを決める」，91年５月号「いまどきのアセンブラプログラミング入門」，91年７月号「C言語によるデバイスドライバ作成法」，91年10月号「80286，80386のアーキテクチャ」「プログラム実行環境『P386』の作成」

①機器に組み込む制御用のソフト（ロボットのモータ制御，OSなど分野としては広範囲），クロスアセンブラ（68K，Z-80など）とそれに付属するリンカ

② MS-DOS

③ PC-98XA

④アセンブラ，C

⑤アセンブラ：MASM，自作のクロスアセンブラ，C：Microsoft C

⑥しない

⑦自作のスクリーンエディタ，MIFES

⑧SYMDEB

⑨イ）使っている

⑩ファイルのアップデート，ページプリンタへのドキュメント清書，ソート／マージなど（すべて自作）

⑪PC上で大規模なソフトウェアを開発してい

● Cコーディング診断システム
Cソースコードの移植性／保守性を診断
Cで書かれたプログラムは移植性が高いといわれているが，実際にはコンパイラやハードウェアに依存する部分が存在し，移植性が損なわれていることが多い．また，Cのコーディングにおける自由度の高さが仇となり，ややもすれば読みにくい，理解しづらいコーディングをしてしまうことも少なくない．
Cコーディング診断システムは，このようなCのソースの移植性・保守性を高めるために，プログラマが記述したCソースコードをテストしてくれるシステムだ．
使用方法は簡単で，MS-DOSのプロンプトから「cchk［オプション］<ファイル名>」とするだけである．オプションには各Cコンパイラの特殊キーワードを回避するものも備えられており，さまざまな診断を行うことが可能である．診断の基準はANSIおよびK&Rに準拠している．
また，パッケージにはユーザーズマニュアル以外に「移植性・保守性向上のためのCコーディングの手引き」，「誤りやすいCコーディング集」が添付されている．移植性の高いコーディングを行わなければならないプログラマにとってはおおいに参考になるだろう．PC-9800シリーズ用のほかにUNIX版もある．
製品名：Cコーディング診断システム／価格：58,000円(PC-9800用)，110,000円(日立2050用)，198,000円(ソニーNews・Sun-3,4・HP9000用)，500,000円(日立E-7500，7700

る人には米国Pocket Soft社のリンカ，RTLink*1をおすすめします．
⑫ツールは，使用頻度の高いものであるから，個人の好みに応じてカスタマイズできるものが有用と思います．
⑬ソフトウェア歴は18年，その間アセンブラを中心にハードウェアを直接制御するような仕事を多く手掛けました．
＊1 RTLink：メモリマネジメント機能を備えたリンカ（（株)エージーテック TEL 052-951-2706）．

足立 高徳

91年5月号「プログラマの仕事場」（株）ミワシステムズコンサルティングの第1技術部取締役部長

① GUIをもった業務用ツール，言語ソフト
② UNIX，MS-DOS
③ SONY NEWS
④ C++，awk
⑤ C++：MIWA C++ + gcc/cc，awk：jgawk*2（インタープリタ）
⑥利用しない．コマンドラインで打つのが好きだから．
⑦ UNIX：vi，MS-DOS：Vz
⑧なし
⑨イ）使っている
⑩ ibuild (interface builder)，その他自作のscript
⑪jgawk（GNU awk）
⑫ UNIXでいう一般的なツール，つまりほかのツールと協力して豊富なバリエーションをもたせられる強力なツール群．
＊2 jgawk：GNUawkの漢字対応版

石野 恵一郎

91年4月号「プログラマの仕事場」（株）ソフトヴィジョンに勤務

① DOSあるいはWindows上AP
② MS-DOS（最近はWindows）
③ PC-H98，PS/2，ほかAT互換機など
④C，C++，アセンブラ
⑤ Microsoft C，Turbo C，Borland C++，MASM，Turbo Assembler
⑥しない．おしきせの環境になじめない．慣れたツールを使いたいから．
⑦おもにVz
⑧ Turbo Debugger，MultiScope*3
⑨make：ロ）使っていない．代わりに自作プログラムを使用．
　linker：イ）使っている．
⑩ノートンユーティリティ*4，fd，ls*5，grep，jgawk，bu*6など
⑪自分がいいと思っても他人がいいと感じる保証はないので，人にすすめることはあまりしない．自分の使うツールは自分で探しなさい．
⑫自分の趣味や，考え方によく合い，便利さを自己主張していないもの．体になじみ，それなしでは生きていられないようなもの．あるものは使う．なければ作る（比較的小規模なら）．
⑬まあ，人のいってることは眉に唾つけて聞くことですね:-)
＊3 MultiScope Debugger for Windows：Windows3.0用デバッガ（MultiScope，Inc.1235 Pear Avenue Mountain View，CA 94043）
＊4 ノートンユーティリティ：ディスク・ファイル管理ソフト，38,000円（ソフトウェアジャパン TEL 03-3479-7151）．
＊5 ls：ファイル一覧表示ツール．
＊6 bu：更新されたものだけをコピーするバックアップツール．日経MIXのk3/pdsに登録されている．

井戸 禎光

創刊号（91年1月号）～「パソコン博物館」を連載中

①～⑫なし．
⑬自己紹介：ソフトを始めてから29年，ただしあまり時間を集中して使っていない（ALGOL，FORTRAN，各種機械語，BASICなど）．ハードをはじめてから25年，こちらの方は趣味と自動制御用コンピュータ（GE-PAC，1800，S-7，JobMaster，270/10，H-10，M-350，FP-1100，9801/F/VM，IFCOM-7，MAXY/note386など）．ただし物によりよく壊れたものは詳しくなり，何となく動いているものはさわれない．新設計をしたものは少なくほとんど改良ばかり……システム屋（悪くいえば評論屋）．

コメント：ソフトの特徴は「少し時間をかけると少しわかる」．したがって「時間を食う底なし沼」に思える．天才的にできる人は身体が丈夫で意志が強く，途中でやめて休める人．そうでないと寝る時間がなくなり身体をこわす．またパーフェクトをねらいたいと思う人は生産性が低くプロにはならない．

ハードの特徴は「すべてわかるか，まったく何もわからないか」にあってとっつき難い．昔のハード屋はシンクロのファンの音とハンダ・ヤニの臭いが好きな人．いまのハード屋は量産部品の選択屋に近く，キーボードでシュミレーション・ソフトを回している人（昔のハード屋はセンサー屋になるのでしょうか？）．

用）／問い合わせ先：日本データスキル（株） TEL 045-641-1211

● C DATA SCOPE
ファイルダンプユーティリティ

プログラミングを行う際，必ず必要になるのがテストデータの設定，確認，ダンプなどの作業だ．C DATA SCOPEは，このような作業を効率よく行うためのダンプユーティリティである．

C DATA SCOPEは，定義ファイルに直接書かれたレコード定義かインクルードファイル中で宣言された構造体を使用して固定長のデータファイルをレコード単位でダンプする．ここで，データの内容を変更したり初期値を設定することが可能である．

C DATA SCOPEで扱うことができるデータファイルは，Cで扱うデータファイルおよび一般の固定長データファイルである．

また，C DATA SCOPEには簡単なデータエントリ機能が備わっており，直接C DATA SCOPEを用いてデータファイルを作ることも可能である．

製品名：C DATA SCOPE／価格：22,800円／問い合わせ先：アイセル（株） TEL 03-3988-6973

●COMPAL/C Light
構造化チャートジェネレータ

プログラミングを行ったときには必ずと

海老原 浩之

91年３月号「私が勧めるこの言語―COBOL」，91年６月号「メモリを気にせず使えるOS/2上のPWB」，91年７月号「PRODUCTS REVIEW：Microsoft BASIC Professional Development System Ver.7.1」，91年11月号「Visual-BASIC」，91年12月号「MS-BASIC7.0」

①OA業務アプリケーション
②PTOS，OS/2，MS-DOS，OEとしてMS-Windows 3.01
③N5200モデル05mk II，IBM PS/55モデル5551V0B，NEC PC-9801RA21
④COBOL，BASIC，C
⑤COBOL：PTOS-COBOL5コンパイラ，COBOL/2，BASIC：MS-BASIC V7.1，Visual Basic，C：Microsoft C
⑥BC7とCではPWBを利用する．理由：OS/2において使用できる統合環境として結構便利である．
⑦MIFES
⑧Microsoft C，BC7共にCV
⑨イ）使っている．
⑩～⑫別になし
⑬自己紹介　氏名：海老原浩之，本業：オーダーソフトの設計，開発を行っている．おもにOA業務が中心

大野 元久

91年３月号「私が勧めるこの言語―C++」，91年６月号「Turbo C++/Borland C++のIDE」，91年11月号「Borland C++」，91年７～12月号「実践C塾　基礎テクニック篇」

①基本的には，個人プログラマなのでPC-9801シリーズ/DOS用のツール類が多い．小さなものでは，プリンタドライバ，ハードコピーツール，ヘルプテキストの表示ツールなど．アプリケーションレベルでは，家計簿などを作成したことがある．細々としたツールはいろいろ．
②MS-DOSあるいはMS-Windows.
③個人では，PC-9801RA2（仕事では，PC-H98 model 70）．
④C/C++と80x86系アセンブラ．
⑤C/C++：Borland C++，アセンブラ：Turbo Assembler.
⑥統合環境を利用する／しないは半々．DOSプログラムでは，エディタ（Vz）＋コマンドラインコンパイラ（BCC）を使うことが多い．Windowsプログラムでは，Borland C++のIDEを使うことが多い．
理由：コンパイルのために使用できるメモリが多いからというのが最大の理由．DOSでもBCXを使えばよいのだが，そのために外さなければならない機能が多く，日本語入力なども考えるとデメリットの方が大きい．また，統合環境は起動するために時間がかかるので，動作を確認する程度の小さいプログラムであれば常駐させておいたVzエディタでプログラムを作成し，コマンドラインコンパイラ（BCC）でコンパイルすることも多い．
⑦Vzエディタ，あるいはBorland C++のIDEの内蔵エディタ．
⑧Turbo Debugger（Turbo Debugger for Windowsを含む）．
⑨イ）使っている．
⑩バージョン管理ツール，（エディタ以外の）ドキュメント作成ツールは，現在，使用していない．ステップ数カウントには，自作ツール（wc）を使っている（単にプログラミングした行数を数えてみるだけで，ステップ数に依存したプログラミングを仕事として受けることはない）．画面制御は，自作ライブラリ（エスケープシーケンスを利用したコンソールライブラリなど）かコンパイラが提供しているものを使う以外の高度なものは使っていない．ただし，将来はTurbo Vision*7（Borland製）を使う予定がある．
⑪［市販品］
・Vz Editor：プログラム，文章の作成・編集のために．単にエディタの機能ばかりでなく，常駐して出力を取り込んだりヒストリー機能を使える点やファイラーが魅力．
・Turbo Debugger：プログラムをデバッグするための必須アイテム．多機能であることはもちろんだが，それらが使いやすい形態でまとめられている点が便利．286や386のプロテクトモードを使って大きなプログラムをデバッグできる点も魅力．
・Resource Workshop ＊8：Windowsのリソースを編集するための必須アイテム．リソーススクリプト（.RC）はもちろん，バイナリリソース（.RES）や実行形式（.EXE）のファイルを扱うことができることや，メニューや文字列などすべてのリソースを一括して編集できる点が魅力．
［フリーソフトウェア］
・lha（吉崎栄泰氏作，ファイル圧縮／展開ツール）：ファイルを圧縮したり，複数のファイルをまとめておくためのもので，パソコン通信においては次のISHとともに必須アイテム．ハードディスクのバックアップをとる場合にも便利．
・ish（M.Ishizuka氏作，バイナリ←→テキストコンバータ）：バイナリファイルをテキストファイルに変換し，パソコン通信のメッセージとして掲載したり，メールでバイナリファイルをやりとりすることができる．
・dam（MASSAN氏作，常駐ツール）：電卓，ファイルビュアー，オンラインヘルプなど，すでにさまざまなサポートファイルが用意されているが，なんといっても自分で常駐機能を拡張できる点が魅力．
・ezkey（c.mos氏作，常駐ツール）：テキスト画面の退避・復帰，グラフィックス画面のク

いっていいほど，ドキュメントを作成しなければならない．しかし，プログラマにとってこれほど苦痛な作業もない．COMPAL/C Lightは，このドキュメント作成作業を支援してくれるツールである．

COMPAL/C Lightは，Cのソースを読み込んでコンパル図と呼ばれる構造化チャート図を生成する．構造化チャート図とは，フローチャートの欠点を改良し，よりアルゴリズムを解読しやすくし，またただれが記述してもほぼ同じチャート構造になるように考案されたものである．コンパル図は，プログラム構造の表現に優れているため，そのままドキュメントとして用いることも可能になっている．

製品名：COMPAL/C Light／価格：18,000円／問い合わせ先：富士ソフトウエア（株）　TEL 03-3455-7876

●CTREE

関数構造図作成ツール

CTREEは，Cのソースコードを解析して関数の構造図を表示するものである．操作は簡単で，まず解析したいCソースコードのあるサブディレクトリに移動し，MS-DOSのプロンプトから

A:¥>CTREE［オプション］

とするだけである．すると，このサブディレクトリにあるすべてのCソースファイルを読み込んで関数の構造図を表示する．もしも，ファイルとして構造図を記録したいのであればリダイ

リアなど，ちょっと便利なコマンドが魅力．
⑫エディタでも統合環境でも，会話型の環境をもつものであればそのユーザーインターフェースが優れていることが重要になるだろう．いまのツールは，たいてい多くの機能を備えているので，それをどのように「使わせて」くれるかということが大切だと思う．逆に，もともとの機能が制限されてしまうものは考慮する気にもならない．そして処理は高速に，消費メモリは少なく，というのはわがままだろうか．現実にはすでに慣れた環境をもつ人が多いだろうから，そうした環境に合わせることができるよう，ユーザインターフェースのカスタマイズ機能をもっていることも重要だろう．機能も含め，それらが選択できるようになっていればベストだと思う．
⑬7～12月号でC塾を連載致しました．最近の興味はもっぱら，C++とWindowsに向いています．現在，言語製品に関する仕事に携わっていますが，またお会いする機会がありましたらよろしくお願いします．

＊7 Turbo Vision：DOS上で利用できるクラスライブラリ．Turbo Pascal付属のもの，BC++対応（IBM PC用英語版）がある．
＊8 Resource Workshop：Borland C++，Turbo Pascal for Windowsの日本語版に付属のリソース編集ツール．

尾崎 行雄

91年9月号「フレームバッファ入門」

①コンピュータグラフィックス関係のユーティリティ．グラフィックスファイルのコンバータ．MS-Windows上の簡単なソフトウェア．PC-9801RA21．
② MS-DOS
③ PC-9801RA21
④C，アセンブリ
⑤ Microsoft C，MASM
⑥しない．使いなれたエディタを手ばなせない．環境が重そうで，使用することで作業効率が目にみえてあがるとは思えない（本当のところは，新しいものをおぼえるのがめんどうそうということです）．
⑦やっぱり，Vz Editorです．
⑧ SYMDEBとCodeViewですが，ほとんど使っていません．
⑨イ）使っている
⑩とくにないです．
⑪とくにないです．
⑫(1)それ自身に致命的なバグがない．(2)問題点を迅速にみつけられるもの．(3)人間のたりない部分をおぎなえるもの．

片桐 明

91年2月号「プログラマ登場」株式会社リギーコーポレーション代表取締役社長

①言語（Mind：日本語プログラム言語，Fifth：FORTH系言語），通信ソフト（PCブリッジ：パソコン間通信ソフトなど自社製品の開発（OSはすべてMS-DOS）．
② MS-DOS
③ FMR-70HD
④ Mind Ver.5.1，Fifth Ver.4.3（共に自社製品），MASM Ver.5.1（マイクロソフト社）
⑤ Mind：Mind（ランタイムはMindで記述），Fifth（コンパイラ本体はFifthで記述）Fifth：Fifth（ランタイム，本体共にFifthで記述）
⑥しない．理由：DOSプロンプロモードで多種のツールを利用するため．
⑦ MEDIT(Mind標準装備のエディタ)，自社製品．
⑧とくになし．
⑨ハ）使っていない，ほかの製品を使用している．makeはMind用のものを利用．linkerはMind/Fifth共に不要．
⑩ find：自作のfind，bcopy：ファイルリスト指定のできるCOPY，bmove：ファイルリスト指定のできるMOVE（以上自社製品），lha：ファイル圧縮ツール（フリーウェア）
⑫定型操作を簡略化できるものであれば何でも有用だと思う．機能の大小，速度は必ずしも有用性と直結しないように思う．プログラマ自身が，日常どのような定型操作をしているかによるのでは．
⑬1周年おめでとうございます．

加野島 英渡

91年3月号「私が勧めるこの言語－Mind」，91年5月号「高級言語とアセンブラのプログラミング」，91年10月号「MS-DOSメモリ管理方式の歩み」，91年4～12月号「MS-DOSツールラボ」，92年1月号～「実践C塾 アルゴリズム編」（共著）を連載中

①作っていて楽しいのは言語ですね．通信ソフトも好きです．あとは，簡単な使い捨てのツール．パソコン通信ログの整理とかテストデータの作成用とか．仕事でならばなんでもやりますけど，Windows 3.0用の通信ソフトを作りたいので勉強中です．あと，去年から「マウスでやるこっくりさん」という危ないソフトを考えているのですが，話で終りそうです．みんなでひとつのマウスの上に手を置いて，「こっくりさん，こっくりさん云々」とやるとマウスカーソルが動いて文字を指す（かもしれない）という冗談ソフトなんですが．
② MS-DOSです．OSと呼んで差し支えなければWindows 3.0も使います．
③ PC-386Mです．
④よく使う順に，(1)C (2)Mind (3)DOSのバ

PROGRAMMING TOOL

レクトすればよい．
製品名：CTREE／価格：7,000円／問い合わせ先：光システム（株） TEL 093-642-6139

● Link & Locate++
ROM化コード開発支援ツール
Link & Locate++は，Lattice CおよびMicrosoft C，Microsoft Assemblerを用いて組み込みシステム用ROMコードの開発を支援するツールである．
組み込みシステムでは，各ボードごとにメモリ構成が異なっているが，Link & Locate++を用いることで，プログラマは自由にプログラムコードを配置することができるようになる．
出力可能なオブジェクトモジュールの形式は，インテルMCS-86オブジェクトモジュールフォーマット(OMF)，インテルのMCS-86のHEXオブジェクトファイルフォーマット，テクトロニクス拡張HEXフォーマット(TEKHEXフォーマット)，インテルのOMF形式のバウンドオブジェクトモジュールである．
リンカ以外にも，パッケージにはライブラリアン，フォーマッタ，ダンプ，チェックサムユーティリティなどが含まれている．
製品名：Link & Locate++／価格：98,000円／問い合わせ先：（株）ライフボート TEL 03-3293-4711

● Plink86+
オーバーレイマネージメントソフトウェアシス

ッチ (4)アセンブラ
⑤同じ順序で，(1) Borland C++，LSI-C86 (2) Pure - Mind (3) COMMAND.COM (4) Turbo Assembler
⑥基本的には統合環境というものは使用しません．使い方を覚えるのが面倒だからです．そうしてまで使いたくなるようなものでもありませんし，MAKEとバッチで十分間に合っていますから．使い慣れたエディタが使えないという不満は大きいです．BC++には統合環境のプロジェクトファイルからMAKEファイルを生成してくれるツールがあって，そのために最初の1回だけ使います．こうすればコンパイラのオプションを覚えなくてすむので（なんたるズボラ）．
⑦ Vzです．
⑧性格がずぼらでおまけにおっちょこちょいなので自分でも呆れるくらいバグは出しますが，デバッガというものは使いません．どうしてこういうものが必要なのかわかりません．

ハード的に不安定な開発中のむき出しのマシンならいざ知らず，ちゃんとしたコンピュータのソフトを書いているのに，自分が書いたループが10回廻ったあとで変数の値がどうなっているのかわからない人とか，自分が記述した条件分岐がどっちに飛んでいくのか自信のもてない人とか，自分が定義した関数にどこから飛び込んでくるのか見当がつかない人ってそんなに多いんですか．「バグを取ってくれる」という意味での「デバッガ」というものがあるのならいますぐ使ってみたいとは思いますけど．

私のバグ取りは，(1)エディタ，またはリストでソースを何遍も眺める．(2)複雑なポインタ操作を書いてしまった場合はアセンブラのソースを吐き出させて確認する．(3)運悪く視線を合わせてしまったやつを引きずり込んでロジックのレビューをやる．(4) CPUを疑う．(5)あきらめて飲みに出かける．という順に深まっていきます．わかりきったことを確認するためにデバッガのコマンドを覚えている暇などありません．
⑨イ）使っている．
⑩そういうものは何も使用していません．プロジェクトごとにハードディスクにサブディレクトリを切って，プロジェクトが終わったら丸ごと圧縮してフロッピーにバックアップをしておくだけです．それがそのままバージョン管理になっています．ドキュメントはエディタとFEPを使って書くだけです．ステップを気にするような仕事をしたことがないのでステップカウントツールというものは使いません．ライブラリはコンパイラ付属のものがあればそれを使います．

強いていえばfdというメニューツールはディレクトリの移動やコンパイラの起動などで日常的に使っていますが．
⑪そういう事情ですので何もありません．くどいようですが強いていえばFDというツールですが，これはあまりにも有名なツールですのでお答えになっていないかもしれません．
⑫質問の趣旨をはかりかねますが……．エディタとコンパイラ（またはアセンブラ）それにリンカがあればプログラミングはできます．プログラミングとは道具でするものだという認識はありません．うーん，なんとも特集の趣旨にそぐわない意見です．
⑬創刊1周年おめでとうございます．P.P誌のような（表紙はともかく）硬派の雑誌が1年ももったということは，日本のパソコンプログラマにとって大変喜ばしいことであります．願わくば来年も再来年も，「2年ももったということは」「3年ももったということは」といいつづけたいものであります．

どこに勤めても喧嘩をして飛び出してしまうので，飛び出せないようにと仲間と3人でソフトハウスを作りました．経営者じゃ飛び出しようがないですもんね．10年ほど前に2KバイトのRAMとカセットインタフェースを積んだZ-80のワンボードマイコンでマシン語を打ち込んで喜んでいたものが，いまでは80386に12MのRAMを積み，200Mを超えるハードディスクをつないでそれでも足りないと騒いでいるんですから，この世界の量的な拡大には目を見張るものがあります．しかしはたして質的にはどうなのでしょうか．それを思うとこの世界の末席を汚すものとして猛烈に恥ずかしくなり，それを紛らすためについつい酒に手が伸びてしまいます．決して好きで呑んでいるわけではないのです．

神林 靖

91年3月号「プログラミング言語の仲間たち」，91年4月号「オブジェクト指向Cプログラミング」（共著），91年8月号「オブジェクト指向概念の基礎」「オブジェクト指向プログラミングの用語解説」

①いまは，あまりプログラミングしませんが，ときどき事務処理用のツール（自動電子メールとか，手紙の自動作成とか）を作ることはあります．
② UNIX(厳密には，Seqent Computer社のDynix)
③ Symmetry
④C，Prolog
⑤ Berkley C compiler，SICS Prolog
⑥それほど大規模なプログラムを書かないので，利用していません．
⑦ nemacs，ときどき vi
⑧デバッガは使用しません．
⑨イ）使っている
⑩ awk
⑪あまりツールに詳しくないので
⑫構造エディタがもっと一般化するとよいのですが．
⑬どのような方々が読者なのかな？　と思うことがあります．読者の皆さんとの意見交換の場

テム

PLINK86+は，MS-DOS上で動作するコンパイラ・アセンブラが出力したオブジェクトモジュールをさまざまに配置・リンクできるオーバーレイマネージメントシステムである．

PLINK86+は，自動的にオーバーレイモジュールをディスク←→メモリ間で交換するオーバーレイマネージャを実行ファイルに組み込むため，ソース上でコール設定などを行う必要はない．

また，オーバレイ構造は最大4095（スタックは32レベル）までツリー構造をサポートしており，大規模なプログラムの開発に威力を発揮する．オーバーレイモジュールは，メモリキャッシュにストアされるので，従来のオーバーレイプログラムよりも高速に実行することが可能である．

PLINK86+は，市販されているほとんどのC，FORTRAN，BASIC，Pascal，COBOLなどのコンパイラおよびアセンブラに対応している．

製品名：PLINK86+／価格：98,000円／問い合わせ先：（株）ソーテック　TEL 045-661-7360

● MKS ツールキット
UNIX学習環境構築ツール

MKS（Mortice Kern Systems）ツールキットは，140を越えるUNIX SYSTEM V互換のコマンドをMS-DOS上で実現したUNIX環境構築ツールである．MKSツールキットは，IEEEが

PROGRAMMING TOOL

をもてるとよいですね．プログラミング言語論／ソフトウェア工学が専門のはずが，最近雑用の多い神林です．

熊谷 信太郎

創刊号（91年1月号）～「コンピュータの法律」を連載中

①～⑫なし
⑬35歳の弁護士です．著作権をはじめ，相続，交通事故などの一般民事事件を中心に仕事をしています．読者の皆さんが法律問題でお困りのときは，気軽に相談してください．

国分 秀昭

創刊号（91年1月号）「プログラマの仕事場」（株）イースト第1開発部主任．91年7月号「OS/2のデバイスドライバ」

① OS/2 PM上のアプリケーション，OS/2 PM上のプリンタドライバ，MS-DOS上のシステムソフト
② MS-DOS（DOS/Vを含む），OS/2 PM
③ PC-9801RA21（RAM 4MB増設（640KB+5MB），HDD130MB）：ソースファイルを編集するためにのみ使用，コンパイル／リンクおよびデバッグはターゲットマシンで行う．
④C，アセンブラ
⑤ Microsoft C（C/2を含む），MASM（MASM/2を含む）
⑥まったく使わない．統合環境内ではコンパイル／リンク時にメモリ不足が発生することが多く，エディタも貧弱であり，プロユースに耐える代物ではないから．
⑦ MIFES-98 V5.0．動作がほかのエディタより高速であり，単なるエディタでなくプログラミングツールとしての機能も多く盛り込まれているのでとても使いやすい．
⑧基本的にはほとんど使わない．FORTRANをながくやっていたので，ソースの要所にデバッグ用のコードを入れてしまうことが多い．とくにOS/2のプリンタドライバではこれ以外にデバッグのしようがない．CodeViewをOS/2のアプリケーションではたまに使うが，作るものが規模の大きいものばかりなので，メモリ不足で使えなかったり，また画面表示が乱れたりしてほとんど役に立たない．デバッグ用のメッセージを表示するウィンドウを作り，そこに変数の値などを表示させることが多い．

MS-DOSでは，ほとんどがMASMでの開発なので，おもにSYMDEBを使い，トレースする程度．後はcref.exeが出力したリストを見る．
⑨ほとんどの場合，付属のもの．とくにこだわりがないので，そのときあるものを使っている．
⑩ CTODOC（ソースファイルから関数仕様書を生成．これは強力！），DOCTOC(CTODOCの逆)，CTREEX（TREEが強化されたもの．これのお蔭で工数短縮できました），VREF，STEP，P（どんなソースファイルでも見やすく整形して印刷．とても便利）．以上すべてイースト（株）製品．

ある特定のプロジェクトのためにだけ使えるツールをそのときの必要に応じて作ることがある．
例1）指定された割り込みベクタをフックし，どのようなI/O要求がアプリケーションによってなされたかファイルに記録するもの．
例2）ファイルの内容を1バイトずつプリンタへ送り，レジスタの内容や各種状況を通知するもの．

このほかでもっとも使用頻度の高いのはエコロジーII[*9]．ただし，これはプログラミングのために使うのではないが…．
⑪とくにない．逆に紹介してほしい．
⑫とくになくても平気だがあると開発期間を短縮できるものが有用なプログラミングツールといえるだろう．だから，まず，「プログラマー・フレンドリー」であること．どんなに役にたつものでも，扱いにくかったり，修得に時間がかかるのではツールとしての意味がない．開発効率が逆に悪くなってしまうからだ．

これとは逆に，使う人間に問題がある場合も見受けられる．ツールとは，それを使いこなす人間があってはじめて本来の機能を果たすわけで，逆に人間がツールに振り回されるとしたら，前述の問題点のほかに，そのツールを使う人間が悪いこともあり得る．これはデバッグ時において顕著な傾向で，本来デバッグに最も強力な「ツール」は人間の頭脳であるはずなのに，CodeViewを起動すると，朝から深夜までそのままマウスを動かしているだけというのはプログラマーとして思考停止としか思えない．

つまり，ツールを使いこなすということもプログラマーの腕の見せどころだということだと思う．逆に言うと，ツールを有効に使いこなせないのなら，プログラミングの基本からやりなおしたほうがいいだろう．どんなツールでも「猫に小判」では，制作者に対して失礼というものだ．
⑬皆さんのなかでプログラマーになりたい人はいませんか？　イーストでは人材を求めています．イーストに入ればこの世は天国．男のなかの男はみんなイーストにはいって花と散る．イーストへはいろう，はいろう，はいろう！

＊9 エコロジーII：ファイル管理ソフト，28,000円（(株)マイクロデータ　TEL 03-3232-9801）

酒井 法雄

91年12月号「Visual Basic」，92年1月号～「イカすBASICプログラミング入門」を連載中

① MS-DOS or Windowsアプリケーション
② MS-DOS/Windows，UNIX（4.3BSD）

提唱しているUNIX標準規格であるPOSIXに準拠している．

MKSツールキットを導入する際，単にMS-DOS上からMKSツールキットのツールを利用するだけ，というものから，MS-DOSのCOMMAND.COMをMKSツールキットに含まれるKorn Shellに置き換えてほぼ完全なUNIX互換環境を構築するものまで，いくつかのレベルを設定することができる．ユーザーの利用法に合わせて，かなり自由な環境設定が可能になっている．

MKSツールキットには，UNIXの基本的なコマンドからストリームエディタであるsed，パソコンユーザーの間でも最近注目を集めはじめたawkインタプリタ，フルスクリーンエディタのviなども含まれている．これらを使いこなすことができれば，プログラミングの強力な武器になるだろう．

製品名：MKSツールキット／価格：68,000円／問い合わせ先：パーソナルメディア（株）TEL 03-3490-8841

● PVCS

LANにも対応したソースコードバージョンコントロールシステム

PVCS（The POLYTRON Version Control System）は，ソフトウェア開発におけるソースコードのバージョンを管理するバージョンコントロールシステムである．PVCSを用いることで，ソースの変更履歴の出力，過去のバージョ

おもなUNIXコマンド
uncompress　compressで圧縮済みファイルの展開

③ PC-9801RA21, NWS-3870
④C, BASIC
⑤ Microsoft C 6.0 (CC), QuickC for Windows, QuickBASIC, VisualBasic
⑥しない．使い慣れたツールを組み合わせた方が使いやすいから．
⑦ Vz, vi
⑧ CV, dbx*10
⑨ニ）使っていない．make：ndmake*11
⑩ rcs, QCASE:W*12, awk, Tex*13, grep, zoo, wc, sed, less*14
⑪⑩であげたもの
⑫CUIならばUNIX環境がイチバン！GUIはこれから．まだ様子見でしょう．
⑬1度はUNIXを使ってみましょう．パソコンもUNIXライクに変身させたくなります．

＊10 dbx：バークレイ版UNIXのソースレベルデバッガ．
＊11 ndmake：シェアウェアのmake．
＊12 QCASE:W：Microsoft QuickC for Windows付属のCASEツール．
＊13 Tex：米スタンフォード大学で開発された文書整形言語システム．
＊14 less：逆スクロール機能をもつページャ．

桜井 健次

91年3月号「私が勧めるこの言語―FORTRAN」

①私の場合，実験データの処理・解析，および数値計算のプログラムを書くことがもっとも多い．これらのプログラムは，処理の具体的な内容を別にすれば，(1)生データのディスクから読みとる．(2)処理・解析の条件を対話式に決定する．(3)各段階での結果をグラフィックス表示する．(4)最終的に得られた加工データをディスクに保存する．(5)行った処理をレポートとしてプリンタに出力する．のような骨格をもたせる場合が多い．

また，データをプロッタに作図するプログラムも多数作成している．私の場合は，作図プログラムは，上記の解析のためのプログラムと分離している．理由は，なるべくきめ細かく，目的に応じた図面にしあげるたいためで，あまり画一的なフォーマットで図面は作りたくないからである（そのようなものであればハードコピーで足りる）．善し悪しは別にして，「図面の数だけプログラムがある」という状況である．

数はそれほど多くないが，重要な位置をもっているのは，実験装置を制御・自動測定するためのプログラムである．GPIBにより機器を制御し，データを取得する．実験の種類により，プログラムは少しずつ違うが，大きなところはほぼ共通であり，一度作った財産が有効利用できる．

② MS-DOS 3.3C (PC-9801), MS-DOS 3.1 (J-3100)

データ処理には大型計算機を使う場合もあり，これは，FACOM OSIV

③パソコン：PC-9801DS/U2, RA5, J-3100 SGT041

大型計算機：FACOM M770

④⑤パソコン：Turbo Pascal 6.0, N88 BASIC 6.0, MS-FORTRAN 4.1

大型計算機：FORTRAN77 (FACOM版)

⑥パソコンでは統合環境は利用していない．TPの統合環境版はメモリを多くとりコンパイル作業に支障があるため．大型では，統合環境PFDを使用している．
⑦ Vzエディタ 1.56
⑧デバッガはTurbo Debuggerをもっているが，仕事には，使用していない．プログラムの性質上，Write文などでチェックしてゆく原始的な方法でほぼ足りるせいもあるかもしれない．
⑨イ）コンパイラ製品付属のものを使用している
⑩⑪Vzエディタに付属しているvmap*15はメモリ使用状況をチェックするのによく使用する．ライブラリとして，Turbo Pascal Numerical Methods Toolbox*16（ボーランド製），GPIBライブラリ*17（サンライズ社製）は愛用し，改変して使用している．
⑫ボーランドのToolBoxシリーズのように，そのまま使用でき，また改変して別個のものを作成するのにも便利というような無数のソースプログラム集が便利だとは思うが．
あと，作業のほとんどはエディットなので，エディターは最も重要な道具である．
⑬本職は，公務員（研究職）．X線を用いた新計測・分析手法の開発に関する研究を行っている．

（日経MIX ksakurai）

＊15 vmap：Vzエディタ付属のメモリ状況表示ユーティリティ．
＊16 Numerical Methods Toolbox：25,000円，（(株) マイクロソフトウェア アソシエイツ TEL 03-3486-1403）
＊17 GP-IB(DMA)ライブラリ：GP-IBインターフェース入出力ライブラリ，75,000円（(株) システムハウス・サンライズ TEL 052-805-5177）

祐安 重夫

創刊号（91年1月号）～「プログラマの道具学」連載中

①アセンブラ，コンパイラ，文書処理，通信，ネットワーク
② UNIX
③ NEWS
④C
⑤ UNIX Cコンパイラ
⑥しない（MS-DOSはあまり使用しないから）．
⑦ emacs
⑧ dbx
⑨イ）使っている
⑩ rcs, Tex

ンの復活などが簡単に行えるようになる．

PVCSはLANにも対応しており，サーバにプロジェクトデータベースを作成して複数のプログラマによるソースの変更履歴を管理することができる．この機能は，LANがない場合にも特定のディスクにプロジェクトファイルをおくことで使用可能である．

また，任意のバージョン・リビジョンの取りだし，格納，変更履歴のレポート出力，同時変更防止機能，パスワードによるファイルプロテクト機能など，数多くの機能を備えている．

さらに，1つのシステムから枝分かれした複数のプロジェクトの変更履歴の管理も可能である．

製品名：PVCS／価格：98,000円／問い合わせ先：（株）ライフボート TEL 03-3293-4711

● PARTNER

マクロ言語を備えたフルスクリーンソースレベルデバッガ

PARTNERは，Cおよびアセンブラで開発されたプログラムをソースレベルでデバッグするためのフルスクリーンソースレベルデバッガである．

パッケージにはリアルモード版とプロテクトモード版の2つのPARTNERが含まれ，リアルモード版はEMSを，プロテクトモード版はプロテクトメモリを利用することで，リアルモードのデバッグ領域をほとんど使用しない．このた

清十郎

91年5月号「逆アセンブルの楽しみ」

①ソース・ジェネレータ，ファイル管理ソフト，機械語エディタ，漢字コード・コンバータ
② MS-DOS
③ EPSON PC-286NF，PC-386S
④80X86アセンブラ，C言語，BASIC
⑤ Turbo Assembler，Turbo C，Borland C++，N88-BASIC(86)(MS-DOS版)
⑥あまりしない．重いので．
⑦ Vz
⑧ SYMDEB，Turbo Debugger
⑨イ）使っている
⑩とくになし
⑪zz386（マイクロオーグ）
⑫小さくて速く，しかも知りたい情報は的確に必要なものだけ表示する．
⑬情報はよく噛んで，食べ過ぎに注意しましょう

谷口 典生

創刊号（91年1月号）～「プログラム『遊』工房」を連載中

①個人的な用途に使うツール類，アイデアを思い付いて作りたくなったもの（dcpは現在フリーウェアとして配布），各種ゲーム．
② MS-DOS
③ PC-9801DA
④C言語，アセンブラ
⑤ Microsoft C，Borland C++，OPTASM
⑥利用しない．現在の環境に比べ，統合環境に利点を感じないから．
⑦ Vzエディタ
⑧ SYMDEB
⑨ make：ニ)フリーウェアを利用している(gmake)
linker：イ)Microsoft C付属のものを利用している
⑩特別なツールなどは使っていません．いつも使うものをあげれば
history*[18]，xscript*[19]，touch，(e,f)grep，gdiff*[20]，less，whereis，sed，jgawk，など．
⑪ツールではありませんが，シリアルドライバならSIAMがおすすめです．
⑫使い慣れたエディタ（高機能である必要はない）と速いマシンがあれば満足です．ツール類は基本的なものがあれば十分．特殊な用途に使うものは自分で作ったほうが早い(?)．
⑬「プログラム『遊』工房」を連載している谷口典生です．毎回ネタを考えるのに苦しみ，原稿が遅れて編集の方には大変迷惑をかけています（ごめんなさい！）．なにか取り上げて欲しいものや要望があればお聞かせください．これからもよろしくお願いします．

＊18 history：岡俊行氏作のコマンドライン編集，入力履歴編集などの機能をもつユーティリティ．
＊19 xscript：岡俊行氏作の出力履歴機能をもつユーティリティ．
＊20 gdiff：GNUのdiff．

田村 浩二

91年3月号「私が勧めるこの言語－Hypertalk/HyperCard」

① Mac ←→ MS-DOS，Windowsのデータコンバート．お行儀の悪いアプリケーションのパッチあてツールなど．その場で必要になったものを作っていく．
② MacOS（漢字talk 6.0.7）
③ Macinosh（SE/30）
④C，Pascal
⑤ Think C，Think Pascal
⑥する．統合環境の使い勝手がよいから．とくにThink Pascalのエディタは Pascal言語専用に作られているので使いやすいため．
⑦コンパイラに付属のエディタ
⑧ MacsBug*[21]
⑨イ）使っている
⑩ ResEdit
⑪Macintoshなら ResEdit
⑫プログラムを半自動的に生成してくれるツール．プログラマは知的な作業ぽに行い，「定石」みたいなものは，機械にまかせられるもの．
⑬Macintoshでプログラミングというのも，なかなか楽しいものです．

＊21 MacsBug：Apple純正アセンブラレベルデバッガ，アップルの開発者向け組織APDAから入手できる（アップルコンピュータジャパン（株） TEL 03-5562-6210）

辻村 浩

91年3月号「私が勧めるこの言語－C」，91年4月号「UNIXのC，MS-DOSのC」

①仕事：デバイスドライバ(UNIX用，CPUは各種) 趣味：BBSホストプログラム
② UNIX，MS-DOS
③ SUN-4
④C，awk，perl*[22]，sh
⑤ cc
⑥しない．理由：必要ない
⑦ UNIX：vi，nemacs，MS-DOS：Vz，jelvis
⑧ kdb*[23]，SYMDEB
⑨ make：ニ)使っていない GNU make(フリーウェア)
linker：UNIX付属のもの
⑩バージョン管理：rcs，ドキュメント整形：plain2*[24]
⑪dcf（まじ）

め，大規模なプログラムのデバッグも可能になっている．

また，インディペンデントモードで起動するとMS-DOSのシステムコールをまったく使用しなくなるため，MS-DOSのシステム内やデバイスドライバ内にブレークポイントを設定することができる．

このほかにもマクロ機能，プロファイル機能，マルチジョブ機能などの優れた機能をもっている．さらに，ハードウェアを付加することで，ハードウェアブレーク機能，リアルタイムトレース機能なども可能となる．

製品名：PARTNER-S／価格：43,000円／問い合わせ先：京都マイクロコンピュータ(株) TEL 075-953-0963

● SOURCER

人気のアセンブラソースジェネレータ

SOURCERは，今もっとも人気のある逆アセンブラ，アセンブラソースジェネレータである．SOURCERは8086から80486までのCPUに対応し，MASM，TASM，OPTASMのそれぞれのアセンブラに対応したソースを出力することができる．解析できるファイルは，COMファイル，EXEファイル，デバイスドライバ，SYSやBINなどのバイナリファイルである．また，メモリ上にロードされているプログラムやROM内のプログラムの解析も可能である．

出力されるソースには，使用されているファンクションコールの機能と各レジスタの意味，

おもなUNIXコマンド
unpack　packで圧縮されたファイルの展開

⑫寝ている間に勝手にプログラムをつくってくれる小人さん.
⑬Tsuporone's BBS (052-752-8053) をよろしく!

＊22 perl：文字列処理，ファイル処理，プロセス処理などを行うフィルタ系言語.
＊23 kdb：kernel debugger，UNIXのカーネル開発時に使用するデバッガ.
＊24 plain2：ベタ書き文書をTexのファイルに変換するツール.

虎岩 登

91年6月号「MS-DOSの開発環境／QuickCの使い方」，91年7月号「MS-DOSデバイスドライバの書き方」，91年7月号〜「IC千夜一夜」を連載中

①計測制御用機器のプログラム，組み込み用1ボードタイプのコンピュータがおもだが，それらを統括する部分には通常のパーソナルコンピュータを使う場合も多い．具体的には，工場の生産ラインに組み込む(中間)製品検査装置のような機器．製品の特性を測定し，良否を自動的に検査するようなもの．したがって私が設計するものは，ほとんどが特注品の少量生産でした．

CPUならZ-80とその末裔が多いが，必要に応じて何でも使う．パーソナルコンピュータ側では最近ではMS-DOSマシンが多い．

組み込み用途では小規模なリアルタイムモニタから作成することもあった．ただし，ここ数年はそのような業務から離れている．

遊びでは，楽しい数値計算とかLISPなんかの言語処理系などを書いて喜んでいるときもある．
②開発にはMS-DOSが主体．Z-80系のプログラム作成には，CP/M3.0を使っていたし，現在でもCP/M上の言語処理系などを使うこともある．
③ PC-386
④各種アセンブラ，C，BASIC(プロトタイプ作成用とか顧客の注文による)
⑤ SLR180：HD-64180やZ-80用のアセンブラ，SLRMAC：LSI-Cの出力のアセンブル用，LSI-C80：80系のCコンパイラ，Turbo C：これと次のはご存じですね，N88-BASIC
⑥統合環境のあるのはTurbo Cだが，まずめったに使用しない．エディタに慣れていないのが最大の理由．とくに以前のバージョンでは機能不足だったので，統合環境を使うのが面倒くさくなってしまった．

しかし，まれにヘルプ機能を利用するために立ち上げるときがある(けど，用がすんだらすぐに抜けてしまう)．
⑦ Vzエディタ
⑧Turbo Debugger　ただし，あまり使わない
⑨ make：（ニ）フリーウェアを使用している(GNU make)
　linker：（イ）使っている
⑩組み込み用の機器のテスト用に簡便なモニタプログラムや自己検査プログラムなどが用意してあるけれども，これはプログラムによるツールであって，プログラミング用のツールではないですね．もっとも，これらは組み込みコンピュータのROMの中につねに入れてあって，製品出荷後もスイッチ1つで使えるようになっていますから，トラブルの原因がソフト側にあるのかハード側にあるのかなどの切りわけに使えます．

あとは，顧客との打ち合わせの時に，画面設計の雰囲気の確認や操作手順の確認のためのプロトタイプを示す場合がありますが，そのときはBASICを使ったりします．
⑪すでにおわかりのように，あまりツールを使っていませんので，勧めることもできません．
⑫ツールではなくて部品だといわれるかもしれませんが，やはり自分で書いたルーチンの蓄積がもっとも有用です．ツール不要のプログラミングのためにも，とくに周辺LSIの取り扱いなんかは定型化したルーチンを用意しておけば，組み込みシステム上でのデバッグもかなり楽になります．人によってはICEが最大のツールだと言っているのも聞きますが，私は使わないし．

でも，いろいろなプログラムを必要に応じてツールに使うことはあります．たとえば，プログラム中に必要なある種の定数テーブルを作成する場合は，表計算プログラムを使って定数を計算し，それをファイル出力したものをエディタなどのテキスト処理プログラムを使って言語の構文に合うようになおして使ったりとか．

いまのように，様々なアプリケーションプログラムが容易に購入・利用できるようになっていると，ツールのプログラムを勧めることの必要性は薄いんではないだろうか．たいてい皆さんもっているでしょう，十分に便利なツール類は．要は使い方ということで．

ひとつこれからの期待をもっているツールといえば，最近の統合環境型言語処理系に付いているオンラインヘルプ機能ですね．ユーザー側で内容を書き換えたり付け加えたりできるようになっている．ここに自分や開発グループで作成したルーチンの仕様を登録すれば，かなり有用ではないかと考えます．しかし，このヘルプファイルをきちんと管理するのは，プログラム設計製作と同じ程度の労力が必要なのが残念です．(その1つの試みを本誌何月号だったけのQuickCの記事中で行ってみましたが)．
⑬マイクロプロセッサは，10年以上前から扱っていました．自分で回路設計から組み上げていって，CPUを味わっていました．そんなわけで，ハードウェアもソフトウェアも両方とも守備範囲にしています．

そうこうしているうち，手元にはさまざまなチップが残りました．かなりの部分は自作基板に実装されていますが，モニュメントにするにはあまりに小さな「石」たちですが，それらの

I/Oポートの操作状況，アクセスしているデータ領域の情報などの詳細なコメントがつけられる．

また，解析オプションと定義ファイルを使用することで逆アセンブルソースリストのカスタマイズを行うことが可能である．

製品名：SOURCER／価格：38,000円／問い合わせ先：(株)ライフボート　TEL 03-3293-4711

● ZZ386

エディタライクな対話型逆アセンブラ

ZZ386は，SOURCERなどのようにバイナリファイルを読み込んでその逆アセンブルソースを出力する逆アセンブラとは異なり，エディタのように画面を見ながら逆アセンブル作業を進めることができる対話型の逆アセンブラである．

ZZ386が対応しているCPUは8086〜80486，ファイルタイプはCOM，EXE，デバイスドライバ，EXP（DOSエクステンダを用いるもの）となっている．

日本語にも対応しており，プログラム中の文字列にシフトjisと1バイトカナ文字が使用できる．自動解析時の文字列解析も同様に対応している．

製品名：ZZ386／価格：39,000円／問い合わせ先：(株)マイクロ・オーグ　TEL 03-3375-3306

●ファイル・エディタ

コレクションなんかもはじめてしまいました．それで，本誌の連載を依頼されたらしいのです．

昔のチップ，基板，周辺装置などの寄付がいただけるなら，謹んでお受けします．残念ながらあまり大きなもの，重いものは住宅事情のために無理なのですけど．

当分の目標は，来年がインテル社の8008発表後20周年ですので，それを記念して，ミニコンタイプのパネルの付いた8008の能力をひと通り引き出せるようなコンピュータを作ることです．

中嶋 雄一

91年10月号「フリーエリアが広がったMS-DOS 5.0」マイクロソフト(株) 技術支援部

①ユーティリティ類：リブートプログラム（98用），ファンクションキー変更プログラム（98用），ディスクコピーツール(汎用)，スタートアップドライブ表示ユーティリティ（MS-DOS5.0汎用）．この程度の小さなものばかり少々……
② MS-DOS 5.0
③京セラ386AX or PC-9801RA21
④アセンブラ，C
⑤アセンブラ：MASM Ver.6.0，C：Microsoft C Ver.6.0
⑥する．CVなどのデバッグをするとき．
⑦ Final(98)，Vz(AX)
⑧ debug.exe，cv.exe
⑨イ）使っている
⑩社内専用バージョン管理・ソース管理ツール(非公開)
⑪sr（sourcer)
⑫できるだけ機種依存しないこと（ライブラリ自体はオプションスイッチで選択する)．たとえば，MASMはMS-DOSマシンであれば98でもAXでも動作するように．
⑬MS-DOSバージョン5.0では，過去のバージョンでは「隠しファンクション」(非公開) になっていたものが少し公開されています．プログラマの方はぜひこの公開されたファンクションを含め，MS-DOSバージョン5.0対応のプログラムを作ってください．

中村 正三郎

創刊号（91年1月号）～「Book Review」連載中

① DOS用ビジネスソフト，ユーティリティ(通信ソフト，エディタ，フォーマットなど)．Windows用ビジネスソフト，ユーティリティ．
② DOS，Windows，MacOS
③ PC-98，PC-H98
④C，C++
⑤ C：Microsoft C＋Windows SDK，C++：Borland C++
⑥利用しない．使いなれたツールを利用したいから．
⑦ Vz，(Brief)
⑧ Turbo Debugger，Turbo Debugger for Windows，MultiScope
⑨ make：イ）使っている
　linker：ロ）使っていない．石野氏作のimake．
⑩ bu：新しい日付のものだけバックアップ，Resource Workshop：Windowsのリソース編集，sgf*25：フォーマッタ，プログラムリストを印刷，ドキュメント印刷，Mac Project*26：プロジェクト管理，Excel：作業時間管理
⑪Borland C++（Turbo Debugger for Windows，Resource Workshopも含む)
MultiScope（Windows用のデバッガ)
PC-AT(互換機) 98より環境がいいです．
⑫漠然としてますが，気がきいているもの，速いもの，安定しているもの，カスタマイズ可能なもの，ソースがあるもの．
⑬DOSやWindowsのソフトをおもに書いているプログラマです．私でもプログラマとしてやっていられるのですから，本誌読者なら元気いっぱいいいプログラムを書けると思います．The BASIC誌に電脳騒乱節というのを連載しています．単行本をお買い上げくださった皆様，ありがとうございました．

＊25 sgf：中村氏作成のSGMLのサブセットフォーマッタ．

＊26 マックプロジェクトII：プロジェクトの管理・報告書作成機能をもつ．87,000円（システムソフト（株） TEL 092-722-4857)

西川 和久

創刊号(91年1月号)～「西川和久の希望的観測」連載中，91年2月号「Microsoft Windows Device Development Kit(DDK)」

① DOS/V，Windowsアプリケーションおよびドライバー
② DOS/V＋日本語Winodws3.0＋NetWare 3.11(US version)
③ PC互換機i80486/33MHz，Cache 64KB，RAM 8MB，HDD 120MB，FDD 1.44MB/720KB＊1，DISPLAY NANAO T560i＋SuperVGA ET4000(1280x1024)
④アセンブラ，C言語
⑤(1)おもにMASM 5.1，Microsoft C 6.0．(2)たまにBorland C++(3)最近QuickC for Windowsで遊んでいる．
⑥しない．HiText(100x37 Vz Editor)が使えないから．
⑦ Vz Editor，MIFES
⑧ Turbo Debugger，PeriScope*27：共にRemote Debug，CodeView for Windows：Windowsアプリケーション Debug
⑨イ）使っている．Micorsoft C付属 nmk

開発環境メイキングエディタ

Cでプログラミングをする際のファイル入出力用のデータ定義部分を，画面上で表を作るようにして簡単に作成できるツール．最終的に完成したデータ定義テーブルからCのソースコードとデータレイアウトのプリントアウトが得られる．

また，データ定義部のネストは5まで可能となっている．共用体のテンプレート宣言は，サポートしていない．

時間がかかるだけで，おもしろくもない作業を短時間に片付けてくれるツールといえる．

製品名：ファイル・エディタ／価格：18,000円／問い合わせ先：エフエーシステムエンジニアリング（株） TEL 0899-31-2886

●簡単ツールエディタ

支援ツールメイキングエディタ

簡単ツールエディタは，エフエーシステムエンジニアリング（株）が発売しているC言語用ライブラリ「簡単シリーズ」をより簡単に使用するためのエディタである．

「簡単シリーズ」に含まれる関数を自作プログラムで利用する際にはさまざまなパラメータの設定が必要になるが，簡単ツールエディタはこれらのパラメータをワープロ感覚で入力し，イメージを確かめながら変更・修正することを可能にしている．設定したパラメータは，Cのインクルードファイルとして利用できる．

簡単ツールエディタには，TEXT，PRNT，

(nmake)/link
ニ) make：gmake (フリーウェア)
⑩とくになし.
⑪PC互換機i80486/33MHz+DOS/V!!最新の処理系がすぐに使えるうえ，速くて安い!! :-) Toolというより環境ですね(^^;
⑫⑪と重複しますが,プログラミングに必要なのは，ツールも含めたトータルな環境だと思います．ベースとなるハードウェアが亀さんだと開発する気になれません．とくにWindows関係だと……
⑬いつも「西川和久の希望的観測」ではお世話になっています．過激な内容やIBM-PC系に話が片寄るのは，個人的な趣味です……お許し下さい (^^;

＊27 PeriScope：IBM PC, PS/2およびその互換機用のデバッガ(The Periscope Company,Inc 1197 Peacgtree St.,Atlanta,GA30361,USA・404/875-8080 FAX404/872/1973).

新居 雅行

91年6月号「Macintoshの統合環境とプログラミング」，91年8月号「Macintosh環境におけるオブジェクト指向」

①原稿用のサンプルプログラム:-)
アプリでできないようなデータ処理プログラム，内容に応じてC/PascalやHyperCardなどを使う．「自分だけで使う」ためのツールをHyperCardを使って……
② Macintosh
③ Macintosh
④C，Pascal，HyperTalk
⑤ THINK C，THINK Pascal，HyperCard (コンパイラではありませんが……)
⑥利用します．統合環境しか存在しない……
⑦…… (コンパイラ付属)
⑧…… (コンパイラ付属)
⑨イ) 使っている
⑩ Prototyper＊28，ResEdit＊29，OnlineCompanion＊30
⑪PrettyC＊31というCのリストをPrintするのがGoodだそうですが，私はリストを打ち出さないので使っていません．
⑫便利な機能とは，本来コンパイラにあるべきでしょうね……
⑬プログラミングが必要か不要か (つまり，既存のアプリでことが足りるかどうか) を判断できるようになりましょう！

＊28 Prototyper：Macintosh用コードジェネレータ.
＊29 ResEdit：Macintosh用リソースエディタ.
＊30 OnlineCompanion：Inside Macintoshの内容を表示するオンラインヘルプ.
以上はアップルの開発者向け組織APDAから入手できる (アップルコンピュータジャパン (株)：03-5562-6210)
＊31 PrettyC：シェアウェアの印刷ツール.

菱 博嘉

91年10月号「MS-DOSのメモリ拡張とVCPI，DPMI」

① MS-DOS中心，デバイスドライバ，386/486プロテクトモードプログラム (レベル0サーバーカーネルなど)，アプリケーション (インストーラー，エディタなど)
② MS-DOS(PC-DOS)，Mac-OS/NeXT Step(Mach)
③ IBM PC互換機，NeXT
④C，アセンブラ，(Pascal)
⑤C：MetaWare High-C，Small-Cベース自作，ASM：MASM 5.x～6.x，(Pascal：Turbo Pascal)
⑥絶対にしない．きらい，必要がない，Link-Makeレベルでかえってめんどくさい．
⑦ Briefかemacs
⑧ SYMDEB，インサーキットエミュレータ
⑨ロ) 使っていない．代わりに自作プログラムを使用．ハ) 使っていない，ほかの製品を使用している．make：MS-MAKE，linker：MS-LINK.
⑩その他のツールはほとんど自作です．
⑪Briefエディタ！ これさえあれば，ほかのエディタはいらない．SYMEDEB，RS-232Cクロスケーブルとデバグコードでできないデバグはないと思う (DOS上では)．
⑫しっかりしたMakeとバージョン管理ツール，手になじんだエディタ，よけいな手出しをしないアセンブラ (書きたいコードが書けること)，バイナリコードが意識できるコンパイラ (なれの問題か……)．
⑬論よりプログラム！ (論よりデバッグ，ともいう)

兵藤 嘉彦

91年3月号「プログラマ登場」

①エディタ
② MS-DOS
③ PC-9801RA21+Turbo486SX
④アセンブラ
⑤ OPTASM
⑥しない．コンパイラを使わないから．
⑦ Vzエディタ
⑧ SYMDEB
⑨イ) 使っている
⑩ grep
⑪Vzエディタ
⑫速いもの
⑬Vzエディタ Ver.2をよろしく(^_^;

WIND，GRPHの4つのモードがある．これらは，それぞれ「キー入力が簡単」，「印字が簡単」，「ウインドウが簡単」，「グラフが簡単」用のパラメータをセットするためのモードである．
簡単ツールエディタでは，これらのモードを切り替えながら作業を進めていくことになる．
製品名：簡単ツールエディタ／価格：22,000円／問い合わせ先：エフエーシステムエンジニアリング (株) TEL 0899-31-2886

● RED2
Cプログラミングを指向するエディタ
RED2は，Cのプログラム開発に適した機能をもつフルスクリーンエディタである．
RED2の特徴的な機能としては，複数のバックアップファイルを作成することでバックアップファイルの世代管理が行える，「¥」マークをバックスラッシュとして表示できる，ASCIIコード一覧の表示，漢字コード一覧の表示，16進コード入力などがある．
また，特にCによる開発のために，自動コンパイル機能，カーソル位置の1ワードをもとにANSI Cの標準関数リファレンスを表示するガイダンス機能，Cの関数リストの作成などの機能が備えられている．
これらの機能以外にも，A～Z，0～9までのキーに文字列を登録しておき，それをワンタッチでテキストに挿入するメタキーやダイレクトタグジェンプ，プログラマーズ電卓などの機能がある．

平田 祥司

創刊号（1991年1月号）「プログラマの仕事場」株式会社マイクロフト

①ハードを含む制御系（コントローラ，通信ハンドラなど），CPU：6809，68000系，V25などMS-DOS，UNIXでクロス開発．事務系ソフトいろいろ．
② MS-DOS，UNIX
③ PC-9801，Sun Sparc station 1++
④C，C++
⑤ Borland C++
⑥しない．Editorが使いなれたものを使いたい．環境が重たい．
⑦ Mifes
⑧ Turbo Debugger
⑨ハ）使っていない．ほかの製品を使っている．Ops Make*32
　ニ）使っていない．GNU make（フリーウェア）
⑩ rcs（PSH4版），Code Base++*33（dBASEファイルアクセスライブラリ），Code Base4.2*34（IBM用入出力ツールとして）
⑪CodeBaseシリーズ，FIXER*35

＊32 OPUS Make：P.35参照
＊33 Code Base++：P.35参照
＊34 Code Base4.2：P.35参照
＊35 FIXER：日本語入力プロセッサ，28,000円（シティソフト(株) TEL 06-927-1060）．

福田 洋

91年3月号「私が勧めるこの言語−LOGO」

① LOGO言語のインタープリタ，タートルグラフィックスによる絵
② MS-DOS
③ PC-9801UV2，NOTE
④C言語，LOGO
⑤ Turbo C 2.0
⑥する：プログラムを作成するとき，エディタとして使用（TC）．
　しない：コンパイル，リンクはメモリ不足から統合環境を使用できないので，コマンドラインから行います（TCC）．
⑦ Turbo Cの統合環境のエディタ
⑧ない．
⑨イ）使っている
⑩ない．
⑪わかりません（プログラムのサイズを小さくして，プリント文を入れるという原始的な方法でプログラムをつくっています）．
⑫プログラムが正しいことを証明してくれるもの．プログラムがループにおちいったときも正しくBVG情報を知らせてくれるコンパイラ．

星野 操

創刊号（91年1月号）〜「98活用アセンブラ技法」連載中

①組み込みシステム
② DOS
③ DA
④C
⑤ Turbo C/Microsoft C
⑥しない．めんどくさい．
⑦Q*36
⑧ PT
⑨イ）使っている
⑩ rcs
⑪FIXER，rcs
⑫頭のいいFEP，現実逃避のできるネット
⑬ WORDYをよろしく．なるべく高級言語を使いませう．ASMはだめよ．

＊36 QEdit：IBM PC用シェアウェアのエディタ．CompuServeなどで入手可．

堀 正明

91年8月号「プログラマの仕事場」株式会社アドミラルシステム　技術部第1課課長

①ディスク管理ユーティリティ：FILE，DIRECTORYの複写，移動，削除などを画面上の一覧より選択し，実行する．
② MS-DOS
③台湾製IBM-PC互換機，PC-9801DA
④ Modula-2，アセンブラ
⑤ Modula-2：Storny Brook Modula-2 V2.2，アセンブラ：Turbo Assembler，OPTASM
⑥最初，コンパイルエラーが出なくなるまで使用し，その後使用しない．メモリ制限などにひっかかりやすくなるため．
⑦ Brief V3.0
⑧ CodeView
⑨イ）使っている
⑩ PVCS：バージョン管理ツール
⑪⑩のPVCS
⑫プログラマが最大限手抜きをできるもの
⑬プログラミングはどれだけ手抜きができるかが勝負だ

宮川 誠一

91年5月号「CPUアーキテクチャとマシン語」

①仕事：リアルタイムOS（自社製・OS名VROW）
［構成］
VES：マルチタスクモニタ，
VIS：I/O規格&規格を満たしたオブジェクト，
BOOT：ブートアップ&ダウンロードetc.，
SBD：デバッガ，その他
　VESとデバッガの一部以外はC言語で書かれています．で，私はVESとデバッガを各

起動時のオプションを設定することでバイナリファイルをエディットすることも可能である．
製品名：RED2／価格：38,000円／問い合わせ先：（株）ライフボート TEL 03-3293-4711

● Right Stuff
viライクなコマンドモードを備えたユニークなエディタ

最近のエディタの多くはモードをもたないモードレスタイプのものになっているが，Right Stuffはテキスト編集モードとコマンドモードの2つのモードをもっている．Right Stuffは，UNIXのviライクな操作性とプルダウンメニューによる現代的なエディタの操作性を合わせもつユニークなエディタである．

テキスト編集モードとコマンドモードの切り替えは「Ctrl+@」キーで行う．コマンドモードを使用せず，通常のエディタとして使用することも可能である．

タグジャンプ，バイナリエディット，文字列・行・矩形でのカット&ペースト，タイプキーマクロなどエディタとしての基本的な機能はすべて揃っている．

また，タイプキー割り付けツールを用いることで，エディタのキー設定を自分の好みに合わせて変更することも可能である．
製品名：Right Stuff／価格：18,000円／問い合わせ先：ソフトウェアビジネスリサーチ（株） TEL 03-3383-4363

CPU用に作っています．現在のところ8086，80386/486(プロテクトモード)，68000，68020用を作りました．計画中はi860，T9000かな．セルフ環境用のツール類とか開発環境の移植とかもときどきやります．

遊び：Mac
最近はハイパーカードのプログラミングが多いかな．３才の子供用に"あいうえお"スタックを作って遊んでいます．文字を書いたボタンを押すと発音するとか読み上げた音の文字を探すゲームとかの構成です．今のとこ．
② MS-DOS (PC9801)，UNIX (HP-UX，SUN-OS)，Mac（漢字Talk6.0.7）
③少し前までは圧倒的にPC-9801でしたが，最近はHP-9000とS4(Sparc)とPC-9801を同時に使用しています．全部ネットで繋がっているものですから．

edit&compile&makeはUNIX側でが多く，仕様書や設計書の入力などの日本語関係はパソコンが多いという使い分けです．

Macは家で細々と使っています．
④C言語，アセンブラ，Hypertalk
⑤C：Microsoft C，TURBO-C++，intel C386，MCC(マイクロテック　X-RAY)，cc (UNIX)，Lattice C(昔のソフトのメンテ時)
アセンブラ：MASM 5.1，intel ASM386，MCC ASM68K
⑥ほとんど使用しない．何処に行っても同じような操作環境の方が良いから．もっぱらedit&makeの環境を使っています．
⑦ MS-DOS：MIFES　V5.0，UNIX：emacs　もうこれがないと仕事する気にならない　:)
⑧ほとんど使わない
必要なときにはそれぞれのコンパイラ用のデバッガを使用する（TC→TD，MSC→CV，etc.)．リアルタイムOSの組込側のデバッグは自作のSDBでやります．
⑨ make(MS-DOS)：付属品は使っていない．いまはiMAKE(石野恵一郎氏＠MIX)ちょっと前までNDmake（iMAKEは現在成長中です．toolbox会議を覗いて見てください)．
make(UNIX)：標準make，GNU-makeどちらも使用しているけれどGNU側に移っている．
linker(MS-DOS)：　付属のリンカを使用．
linker(UNIX)：付属のリンカを使用．
⑩とくになし．強いてあげるならawk，grep，sedは多用しています．
⑪awk，grep，sed，perlはまだ使っていない．
⑫大きなシステムを作成するときには仕様(案)書と設計書を作成する段階で勝負が決ります．この辺をサポートしてくれるCASEツールには期待しているのですがまだ実用的とまではいえないようです．

プログラミングの実作業ではエディタが重要でしょうね．あとは味気ない繰り返し作業を確実に繰り返してくれるための道具類として⑪であげたツールが有用です．

最近はmake&awkでテスト行程を自動化するようにしたりしてバージョンアップ時の負担が減りました．
⑬開発環境は自分で育てて行くものです．パソコン通信で探すのも良いし，自作するのも良いでしょう．与えられた環境に文句だけいっていては良くなりません．

最近GNU関連のツールを使うようになって世界中には素晴らしい人達が沢山居るなーーと実感しています．もう少し英語が解るようになるともっと堪能できるのですがまだ道は遠そうです．

松井 栄伸

91年12月号「プログラマの仕事場」（株）スタジオ・ゲンに勤務

①グラフィックス系ソフト(3Dモデリング&レンダリング，マルチメディア)
② MS-DOS
③ PC-9801
④C，C++，アセンブラ
⑤ Microsoft C，Borland C++，Turbo Assembler，MASM
⑥しない(Turbo Pascalではしていた)．大きなプログラムが統合環境から走らない(走らなかった)．QUIT→実行→再起動は時間がかかるのでしたくない．また，普通使っているエディタと使い勝手が異なる．
⑦ MIFES
⑧ほとんど使わない．printf挿入が多い．
⑨イ）使っている
⑩紙，鉛筆，机（コーディングが終了するまでは，ほぼこれで足りる)．プリンタ用紙(デバッグ以降)．コンピュータ上でやらない理由：１画面に表示される情報量が少なすぎるため．

エコロジー，TAB-スペース変換ツール，あとは必要に応じて作る．
⑪紙，鉛筆，机，エディタ(プログラム打ち込み&デバッグ&ドキュメント用)，ファイル管理ツール(エコロジーなど)これだけで十分．
⑫デバッギングを前提としたツールが多すぎる．最初から極力バグの出ない，またデバッギングしやすいようなプログラムを作るべきである．
⑬自分のプログラムも３日たてば他人のプログラム．分かりやすいプログラムが最良のドキュメントとなります(もちろん本物のドキュメントも大切)．

● Final Super Pack

16機種に対応するエディタパック

Final Super Packは，１つのパッケージでNEC PC-9801/98，富士通FMR/FM-TOWNS/FM-16β，松下電器Panacom M，日立B16/32/2020，東芝J-3100/3300，日本IBM PS/55，IBM PCおよびその互換機，各社AX，シャープX68000に対応している．パッケージにはこれらのマシンに対応したディスクとマニュアルが収められ，ユーザーは自分の環境に合わせてエディタをセットアップすることができる．

複数の異なるマシン上でプログラムを行う場合，使い慣れたエディタを使えず往生することがあるが，Final Super Packはこの問題を解決してくれるだろう．

エディタとしての基本機能に加えて，正規表現を用いた検索・置換，バイナリエディット，短縮文字列入力などの機能を備えている．

また，Finalには，UNIX上で動作するX-Finalという製品も存在している．

製品名：Final Super Pack／価格：28,000円／問い合わせ先：（株）エー・エス・ピー　TEL 03-3767-1451

● VISP

C言語文法チェック機能をもつエディタ

VISPのもっとも大きな特徴は，Cのシンタックスチェック機能を備えている点である．コンパイル前にエディタ内でプログラムの文法チェ

森　誠

91年10月号「DOS エクステンダ活用プログラミング」

①dBASEファイルとテキストファイルの変換ユーティリティやハガキの宛名書きプログラムなど.
②MS-DOS
③NEC PC-9801RA
④C, アセンブラ
⑤Microsoft C, WATCOM C/386, Lattice C, MASM
⑥しない.
(1)エディタは以前から使っているもので使いたい. (2)立ち上がりが遅い. (3)オプションの設定がメンドウ(どのメニューの中に目的のオプショがあったか, 迷うことがある)
⑦RED2(コンパイラ付属以外の一般商品)
⑧CodeView(コンパイラ付属), CPR*37(コンパイラ付属), Instant-D32*38(DOSエクステンダ「DOS/4G」に付属, C-SPRITE*39(一般商品)
⑨イ) 使っている(makeはWATCOM C/386についてくるwmake)
⑩dBASE互換のデータベース・ファイル作成ライブラリ AccSys*40.
⑪16ビットDOSエクステンダーDOS/16M, 32ビットDOSエクステンダーDOS/4G
⑫単機能でまとまっているもの(ビジネスアプリケーションなら, 複雑な機能をとりこんでも使いようですが, ツールは単純がいい). ほかのツールと組み合わせて使いやすいもの.
⑬他誌がプログラマ向けから一般アプリケーションユーザー向けに記事の内容が変わっていく中で, 1年前に創刊された本誌はプログラマにとって読みごたえのある内容を保ち続けるであろうことを期待しています.

＊37 CPR(CodePRode): Lattice C付属のフルスクリーン・ソースレベルデバッガ.
＊38 Instant-D32: DOS/4G付属のプロテクトモード用ソースレベルデバッガ.
＊39 C-SPRITE: C言語指向のシンボリックデバッガ, 現在は発売中止.
＊40 AccSys: dBASE互換のデータベース処理用C言語ライブラリ, 再販ライセンス付き98,00円, ソース付き198,000円(いずれも(株)ライフボート TEL 03-3293-4711)

柳澤　健二

91年9月号「IBM PC のビデオシステム入門」

①一般誌などのコラム書きその他が本業ですので, 自分の仕事もしくはそれ用のパーソナル・コンピュータそのもののための細々としたユーティリティ作成がおもです. 最初に作ったのはクロスワード・パズル作成専用の辞書つきエディタでした. その後, コメント管理などのファイル管理ユーティリティ, 外字作成ツール, テキスト閲覧用のプログラムなど. まあ要するに素人プログラマです.
②MS-DOS(PC-DOS, DOS/V)
③PS/55noteとPS/5530Sです.
④Pascalときどき BASIC
⑤Turbo Pascal6.0(IBM版), QuickBASIC4.5(IBM版)
⑥あまり使いません. 日本語環境では動作しないから. 動けばもうちょっと使いたいです.
⑦MIFES
⑧きちんとデバッグしたいときだけ英語環境で「統合環境」のデバッグ機能を利用しています.
⑨イ) 使っている.
⑩とくにありません. 強いていえば自作の(原稿書き用)ツール類はプログラミング時の環境でもあります.
⑫入手してすぐ役立ち, 1年使わずにいて久々に使うときもすぐ使い方がわかるようなものであることが僕のような「サンデー・プログラマでさえない人」には有用であると思っています.
⑬あけましておめでとうございます.

山本　英昭

91年3月号「私が勧めるこの言語—BASIC」

①情報処理教育関係の教材. ディスク, イメージデータなどの各種のファイルを直接操作する簡単なプログラム.
②MS-DOS
③NEC PC-9801, EPSON PC-286, 富士通FMR-50
④BASIC(N88-日本語BASIC, BASIC/98, F-BASIC86)
⑤付属のコンパイラ(BASICC, BASIC98C, FBASCP)
⑥とくに必要としないので利用していない.
⑦付属のエディタ, MIFES, RED
⑧SYMDEB
⑨イ) 使っている
⑩〜⑫とくになし
⑬最近はプログラミングの時間があまりとれなくなってきたので, できるだけフリーウェアなどを利用し, 不十分なところを補う程度の簡単なプログラムのみを自作しています.

山本　博昭

92年1月号〜「実践C塾　アルゴリズム編」(共著)を連載中

①仕事以外のプログラミングは実はあまりやっていません. 必要に迫られて使い捨てのツールを書くとか, パソコン通信のログ整理ツールを書くとかはやっているのですが……
②ほとんど MS-DOS 1本槍です. OSではないような気もしますが, Windowsは「ソリティア」をやるためだけに起動したりします. パ

ックを行えるので, つまらないミスでコンパイルをし直す必要がなくなり, プログラミング時間が短縮できるだろう.

このシンタックスチェックは, ANSI Cの規格に準拠したものである. エラーを発見すると, エラーのある行の先頭にカーソルが移動し, プログラムを修正することができる. このときにチェックされるのは, あくまでもコンパイル時に発生するエラーのみで, リンク時に発生するエラーを発見することは残念ながらできない.

このほかの機能として, あらゆる削除・インサートコマンドで128コマンドレベルまで可能なアンドゥ・リドゥ, キーボードマクロ機能, 26個の英文字に任意の文字列を割り当ててキー入力を軽減するパーマネントマクロ機能などがある.

VISPには, PC-9801版以外にDOS/V版も発売されている.

製品名: VISP／価格: 38,000円／問い合わせ先: スピリット(株)　TEL 03-3234-8041

● WordMaster

フルスクリーンエディタの元祖

WordMasterは, CP/Mの時代に一世を風靡したフルスクリーンエディタの元祖ともいうべきエディタである. WordStar同様, ダイヤモンドカーソルを備えていることはいうまでもない.

WordMastarには, ビデオモード・コマンドモードという2つのモードがある. ビデオモードはタイプされた文字がそのまま入力されるモー

ソコンを買ったときについてきたBASICは封を切っていません.
③ PC-386MとPC-386NOTE-Wです.
④普段よく使っているのはアセンブラとMindコンパイラです. 必要にかられるとCも使いますが（問題発言）, 一番得意なのは日本語（ただし札幌シティローカルバージョン）です.
⑤アセンブラは「Turbo Assembler」, Mindコンパイラは「PURE Mind PRO」です. CコンパイラはMS-Cを使っていた時期もありますが, 最近はBC++を使っています. 開発スピードが遅いので, せめてコンパイルが速いのが救いです.
⑥原宿の「東郷神社」なら学生時代に時々散歩にいきましたが, ではなくて, コンパイラ付属の統合環境はあまり使っていません. ファイル管理ツール「FD」でエディタとコンパイラとデバッガを瞬時に呼び出せるようにしていますので自前の統合環境と言い張っている状態です.
⑦ Vzエディタを使っています.
⑧古い付き合いのSYMDEBが多いです. どちらかというと机上デバック派なうえに, 紙の上でこんがらがってくると「printfデバッグ」というパターンになっています.
⑨最近はすべてBC++セットになっています. Mindはリンカがいりませんし.
⑩私のパソコン環境は, A.Idei氏作のフリーソフトウェア「FD」を基本に成り立っています. ファイル管理からほとんどすべてのプログラム起動まで, このツールをベースに操作していますので, DOSのプロンプトは1週間に数回しか見ないという状態です. A.Idei氏の住む岡山方面には足を向けて眠れません.
⑪メモリ絡みのデバッグになったら, K-CRAFT氏作の「MS[*41]」や, うのしん氏作の「MEMM[*42]」が是非お薦めです. とくに, 後者は常駐して任意のアドレスの確認や書き換えが可能ですので, 常駐ソフトのデバッグなどに大変重宝しています. これらはすべて, フリーソフトウェアであり, 本当に助けられてばかりです. 足を向けて眠れない人がこのまま増えていくと, 立ったまま寝なきゃならないのではと心配しています.

また, オーバレイなどで動作の複雑なソフトには, ファイルのオープン/クローズの状態を記録する常駐ソフトなどを用意しておくと便利だと思います.
⑫作るプログラムの種類やトラブルの具合によって必要になるツールは変わってくると思いますが……作ったプログラムを見通してバグの在処を嗅ぎつける「推理力」と, その場に必要なツールを探し出す情報力, または自作する能力が第1に求められるでしょう……というのは当たっているようで実はあまり役にたたない意見ですね. 私の場合, 一番有効なツールは旨い肴とおいしい酒ということになっています.
⑬創刊1周年おめでとうございます. プログラマーの端くれとして第1号から読ませていただいています. 今年もますますパソコンをとりまく情勢は混沌としてくるのでしょうが,「プログラムを作る立場」というスタンスから有用な, そして楽しい誌面を期待しております. 毎号楽しみに読んでいた貴誌に, 自分自身の名前が目次に載るということで幸せ半分, 不安半分の状態です. 新米ですが, なにとぞよろしくお願い致します.

私もN-BASICやCP/Mを使って仕事をした微かな記憶があるので, キャリアは相応に長いはずなのですが, 日進月歩のソフトウェア業界の中にいると, 毎日が新人のような気持ちです. 昨年10月から, 3人でソフトハウスを始めまして, ますますこの世界にのめり込んでいく0x22歳です.

＊41 ms：MS-DOS汎用の統合メモリ管理情報レポーター. NIFTY-Serve FGALなどで入手可.
＊42 memm：常駐メモリダンプ&エディットツール. NIFTY-Serve FGALなどで入手可.

若月 文男

91年6月号「プログラマの仕事場」（株）エム・ピー・テクノロジー常務取締役. 91年7月号「Windowsのデバイスドライバ」

①アプリケーション（MS-Windows, OSF/Motif（X-Window））
② MS-DOS, MS-Windows, X-Window（X11Rel4）
③ PC-9800シリーズ, EWS4800/220
④C言語, アセンブラ
⑤マイクロソフト製のもの
⑦ MIFES, vi
⑧ SYMDEB, CodeView, dbx
⑨イ）使っている

（敬称略）

ドで, コマンドモードはラインエディタEDLINのコマンドである行編集コマンドを用いて検索・置換などを行うモードである.

最新のエディタが備える豪華な機能こそないものの, シンプルで使いやすいエディタである.

製品名：WordMastar／価格：38,000円／問い合わせ先：ワードスター・ジャパン（株）TEL 03-3423-4901

● ZED

オーバーラップ型マルチウィンドウテキストエディタ

ZEDは, オーバーラップ型のマルチウィンドウ・マルチファイルのフルスクリーンエディタである.

ZEDには, 複数回遡れるアンドゥ機能, 複数の画面を同時にスクロールできるマルチ画面スクロール機能, 一定期間キー入力がなかったときに作業のバックアップを取るオートバックアップ機能, 画面の焼き付きを防止する画面オートオフ機能などがある.

また, ZEDは1つのファイルを複数のウィンドウに同時に表示して編集することができる.

ZEDはマウスに対応しているので, 編集操作をマウスで行うことが可能である.

製品名：ZED／価格：18,000円／問い合わせ先：パシフィックソフトウェア開発（株）TEL 0888-43-3618

パソコン博物館 XIII

十進法のパソコン FP1100

カシオ計算機 FP1100

井戸禎光

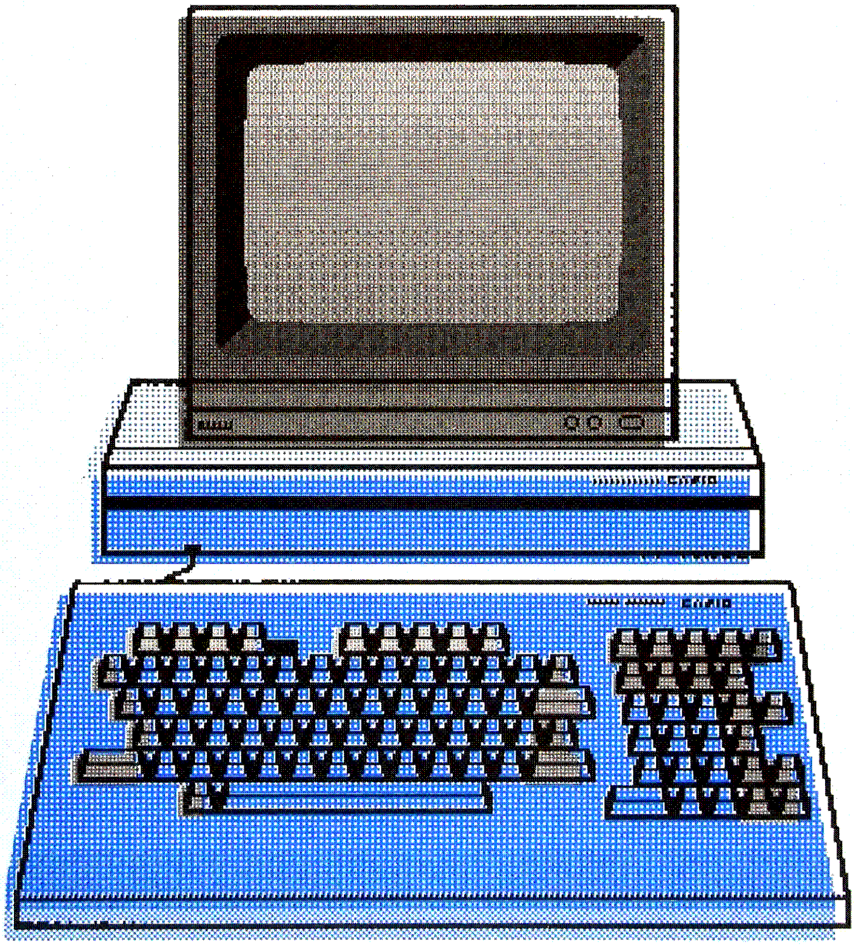


Illustration
TOSHIHITO ITO

電卓のカシオが出すパソコンはどんなものになるのだろうと興味深く待っていた．そこに現れたのが，1982年に発売されたFP1100とモノクロのFP1000である．BCDで演算していたり，角度にしても電卓並にDEG・RAD・GRADの3種類あったり，倍倍精度計算という，指数が十進で2桁，仮数が29桁もある計算ができたり，といった特徴のある機械で，価格はそれぞれ128,000円と98,000円であった．

性能面では，当時価格が2倍近かった機械である PC-8800や FM-8に迫るものをもっていた．その構成は CPU に4MHz の Z-80A（μPC780C），それにサブ CPU（μPD7810G）をもっていて，I/O の操作をまかせ，ペリフェラルコントロールも専用の IC（μPD765AC，μPC8251AC，HD46505など）を多用することにより，コストパフォーマンスの改善に成功している．

サブ CPU の μPC7801G は8085＋PIO といった感じの IC である．おもしろいのは，最近の IC では外部から与えるクロックサイクルを速くしにくいため，内部クロックが外部クロックの2倍といったものが見受けられるのに対し，この IC は外部クロックの半分が内部クロックとなっている．そのためとくに高速を要するグラフィックスなどの CRT 表示はあまり速くはならない．しかし，計算にかけては非常に速く，とくにルートでは他機の10倍も速いことがある．

RAM は，主メモリの64KB とは別に48KB のビデオ用 RAM をもち，トータルで112KB をもっていた．主メモリは，64K ビットの HM4864P-2 150nsec（当時の単価は2800円程度）を8個使っており，ビデオ用 RAM は16K ビットの HM4716AP（当時の単価は500円程度）を24個使って，全ドット RGB8色に対応していた．

外部メモリとしては，フロッピーがだんだん安くなってきてはいたが，未だ高価（両面倍密度×2台で298,000円）であり，標準は未だカセットテープを使うようになっていた．そこで問題は立ち上げ時に起こる．すなわち ROM BASIC で立ち上げることは可能であるが，ほかの言語を使おうとすると困ったことになる．外部記憶装置から立ち上げることが考えられるが，カセットテープの信頼性のなさには泣かされていた．

FP1100の場合は，64K すべてを RAM でもち，BASIC の入る番地の部分に ROM がだぶって格納されている．したがって信頼性は高く，そのうえ高速である．またほかの言語を使いたい場合には，32KB の“ROM パック”を使うことができ，これを使ってブートすることも可能であった．

本機が売り出されると，すぐに筆者は仲間十数人と協議の上，購入したのであるが，当時の選定理由のいくつかを思い出してみたい．

①ソフトの勉強など（たとえば BASIC）の使い道を決めたとき，それに十分な性能をもっている．

②価格が予算以内である．

③他社の製品とくらべて価格が安く，自作するよりも安い．

④ハードウェア，とくにバスインターフェースが公開されていて，十分な情報が入手できる．

⑤ソフトウェアの必要な部分が公開されていて，とくに I/O のエントリポイントが示されている．

⑥メモリマップが公開されている．

以上のような条件を満足する機械はめったに出ないが，当時の FP1100については，ほぼ満足できていた．④～⑥については，『新電気別冊「C82BASIC マスターブック」』，『I/O 別冊「ペーパーウェア」』が出て，少し遅れて『FP1100 100%活用法』も出版された．

FP1100は十進法で計算するので，3で割って3を掛けても0.999にはならないコンピュータであった．残念ながらほかの理由でパソコンのメインストリームにはならなかったのであるが，ひとつの強い主張であった．

●汎用拡張スロットインターフェース

番号	A列（下側）	B列（上側）
1	＋5V	GND
2	＋5V	GND
3	＋12V	＋12V
4	＊PSL(O)	φ(3.9936MHz)(O)
5	＊BSL(O)	＊RESET(O)
6	＊REQ(O)	＊REQD(O)
7	＊RD(O)	＊WR(O)
8	＊CS(O)	
9		
10	＊NMI(I)	＊LPS(I)
11	＊INTA(I)	＊INTB(I)
12	＊INTC(I)	＊INTD(I)
13	D0(IO)	D1(IO)
14	D2(IO)	D3(IO)
15	D4(IO)	D5(IO)
16	D6(IO)	D7(IO)
17	＊A0(O)	＊A1(O)
18	＊A2(O)	＊A3(O)
19	＊A4(O)	＊A5(O)
20	＊A6(O)	＊A7(O)
21	＊A8(O)	＊A9(O)
22	＊A10(O)	＊A11(O)
23	＊A12(O)	＊A13(O)
24	＊A14(O)	＊A15(O)
25	-12V	-5V
26	＋5V	GND
27	＋5V	GND
28		

＊：負信号　(O)：出力信号
(I)：入力信号　(IO)：入出力信号

祐安重夫

プログラマの道具学

第13回 エディタ

Illustration
Mikio Sasaki

現在のようにパーソナル・コンピュータやワークステーションによる，オンラインのソフトウェア開発支援環境が普及する以前は，プログラムはコーディング・シートとよばれる紙に記述し，それをキーパンチャーにわたして，紙カードや紙テープにパンチしてもらうか，自分でキーボードをたたいてパンチするという手順をふんでいた．

カードの場合は1枚が1行に相当していたので，プログラムを修正する場合は不要なものをぬきとり，必要なものを再度パンチして追加していくという手順をふんだ．またキーパンチャーにたのんだ場合は，ちょうど原稿が印刷所で組版される際に誤植がでるように，うちまちがいもかなりの頻度で出現した．

テープの場合は修正の必要なところまで，以前のテープをよみとりながら別のテープにコピーし，修正内容を再度パンチしたり，以前のテープの不要な部分をコピーしないでよみとばすといった作業を，手動で注意ぶかくおこなう必要があった．

どちらかといえばテープよりカードの方が，修正が楽だったといえるが，カードの場合はうっかり床にばらまいたりしてしまうと，収拾のつかないことになるという欠点があった．もっともカードについては，通し番号をパンチするための領域が用意されており，そこに番号をうっておけば，機械をつかって順番にならべかえることが可能ではあったのだが，面倒なときなどはつい番号を省略してしまって，ひどい目にあったこともある．また通し番号はあとからの追加をかんがえて，下1けたか2けたをあけておくのが普通だったが，追加をくりかえすうちに番号がたりなくなるということも，結構あった．

紙テープの場合は，最新のものに常に最新版と明記しておかないと，あとでどれが一番あたらしいかわからなくなる，などという冗

談を真にうけて，最新版テープの山をつくってしまった人がいたという話をきいたこともある．テープにしてもカードにしても，いらなくなったものはゴミとしてすてられるだけだった．

どちらにしても今おもえば，紙資源の無駄づかい以外の何物でもない．もっとも当時はコンピュータ自体が，現在とは比較にならないくらいに高価なものだったので，紙のことなど誰も無駄とはかんがえていなかったようだ．これをおおきくかえたのは，1973年頃におきたオイルショックだったかもしれない．石油と紙不足との関係は，今になるとわかるようでわからないのだが，巷ではトイレットペーパーをもとめて行列ができたし，出版業界は用紙不足に苦労した．コンピュータの世界でも，プリンタ用紙などの節約がさけばれるようになってきた．

別にこのことが理由だったわけではないのだが，徐々にコンピュータが十分な量のメモリやディスク装置をもつようになって，テープやカードにプログラムをいれておかずに，コンピュータの中にファイルとして保存しておくことが可能になってくる．これにオンラインのテレタイプやCRTディスプレイがつながることで，直接プログラムをキーボードから入力することが，かんがえられるようになった．こうしてプログラムの入力を支援するためのソフトウェアとして，テキスト・エディタが出現してくる．

最初の頃はエディタはそれほど高度な編集機能もなく，行を番号で指定してそれを削除したり，あたらしい行を挿入するといった，カードの編集をオンラインにした程度のものだった．その後，行単位ではなく文字単位に編集できるものが出現し，CRTディスプレイの普及とともに，画面上に編集対象となるプログラム・テキストを表示しながら，直接位置を指定して挿入や削除を実行できる，スクリーン・エディタというのが，あたりまえのことになっている．

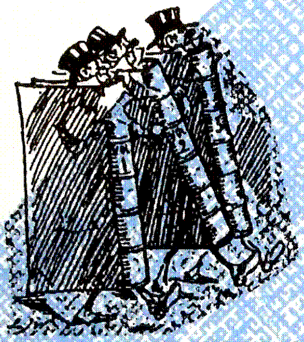
最初に使用したスクリーン・エディタは，8ビットのマイクロプロセッサである8080やZ80用のOSであったCP/Mで動作するWord-Masterというソフトウェアだった．これはじつに感動的な体験で，このような便利なものともっとはやくであっていれば，仕事が楽に効率よくできたのにとおもったものだった．いまではスクリーン・エディタの存在はあたりまえのことになっており，それなしではプログラミングも原稿の執筆もできないところまできた．

IC千夜一夜

第7回 初期のパソコンをリードしたCPU

MCS6502

虎岩 登



ハードウェアアーキテクチャ

MC6800に準拠した設計

今回取り上げるMCS6502は，古くはApple IIやPETなどに使われていますし，最近ではファミコンのCPUにもこれに手を入れたものが使われたことで有名です．もしかすると，プログラミングの経験のある方もいらっしゃるかもしれません．

MCS6502はMOSテクノロジー(MOS technology)社の製品で，1975年ころに発表になっています．同社からはこれ以前にMCS6501というCPUも出荷されています．が，MCS6501の方は先月取り上げたMC6800のセカンドソースです．このような経緯からもわかるように，MCS6502は，MC6800とある程度の類似性をもっています．中でも最たるものはピン配置に表われているといえるでしょう．図1にピン配置を示しますから，先月号のMC6800のピン配置と比べてみてください．クロック信号のϕ1とϕ2は出力信号になっていて機能が異なっていますが，異なった信号はこれを含めても8本だけで，それ以外の32本は名称・機能とも同じです．また，多少の外付け回路で，バスタイミングもほとんどMC6800と等しくすることが可能です．

当時のマイクロプロセッサは，入出力はTTLレベルといっても，クロック信号だけは特殊な条件を要求されたものです．当時の技術的制約から，チップ内部の多数のトランジスタを直接ドライブしていました．ですからMC6800ではほぼ0Vから5Vまでをフルスイングする信号が必要で，その負荷容量も他の入出力端子の10倍以上を要求していました．8080にいたってはクロック信号のハイレベルは9V以上が必要でした．どちらも専用のICか，特別な回路でドライブすることになっていたのです．この点が，MCS6502では改善されて，普通のTTLレベルの信号をϕ0端子に入力すると，外付け回路用の基準信号であるϕ1とϕ2を作ってくれます．

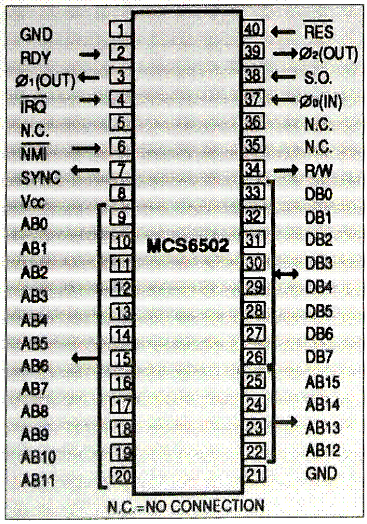
バスタイミングがMC6800とほぼ同じということは，すでに開発されているMC6800用の周辺LSIをそのまま利用可能ということにもなります．半導体メーカーの立場からすれば，競争相手と同じようなLSIをひと通り開発する投資を減らすことができると同時に，最初から競争相手と同じくらい豊富な周辺LSIを利用可能であると宣伝できるわけです．ユーザー側では，ハ

ードウェア開発の経験や開発済みの周辺回路を生かすことができます．場合によっては，MC6800を使ったコンピュータのハードウェアをそのまま使って，CPU部分だけ置き換えることも可能でしたから，ハードウェア開発の期間や手間が少なくなる分有利になります．もっとも，命令セットはMC6800とはまったく異なっていますから（アセンブラレベルでは多少似ているところはありますが），そのまま使うということはできません．

6800系のバス・タイミングのようなものは公式規格にこそなっていませんが，この業界で標準になっている規格というのは多かれ少なかれこのようにメーカー間の利害が絡んで標準化が行われるという経緯があるものです．MOSテクノロジー社はインテルやモトローラなどと比べれば，はるかに小さな会社でしたから，ある種の「事実上の標準規格」に準拠した製品を出して，開発力の小ささを補ったのです．

レジスタセットや命令群を省略
10進演算は容易に

図1　MCS6502のピン配置

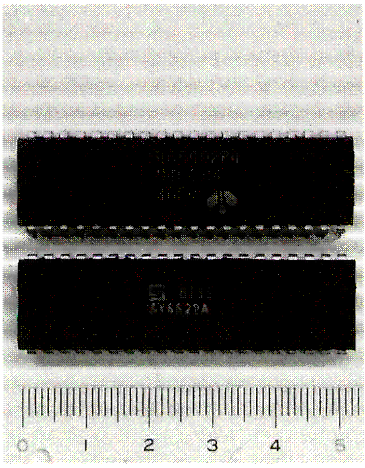


ソフトウェア面から眺めると，この時期のCPUとしては２つの特異な点があります．あまりにも大胆に切り捨てられたレジスタセットや命令群と，間接アドレッシングまで含む豊富なアドレッシングモードです．

レジスタセットを図２に示します．MC6800と比べると，インデックスレジスタの数が多くて，一見しただけではどこが省略されているのかと不思議に思われるかもしれません．せいぜいアキュムレータがひとつ減っているだけのようにも見えます．しかし，よく見ると，インデックスレジスタとスタックポインタのビット数が8bitになっています．

したがって，インデックスレジスタの使い方は，MC6800のように16bitのインデックスレジスタに8bitの数値を加算して対象アドレスを決定するという使い方ではなくて，16bitの数値を8bitのレジスタで修飾する形になります．この方式で256Byteを越えるデータの転送をしようとすれば，インデックスレジスタをインクリメントするだけでは追い付かなくなってしまうのは明らかです．この問題に対する解答としては，複雑な修飾関係のある間接アドレッシングがいくつか用意されています．

写真1　C-MOS版高速6502(上)と，6502用の複合型入出力LSI6522(VIA)(下)
残念ながらオリジナルのCPUはもっていません(著者)



スタックポインタも8bitしかありませんから，同様の制約が生まれます．スタックポインタは，16bitのアドレスのうち下位8bitだけを表すことにして，上位8bitは16進数の01に固定されています．256Byteを１ページという単位で数えることにすると，64KByteのアドレス空間は256ページから構成される計算になります．そこで最初のページ，つまり０ページは，MC6800のベースページと同じように8bit長アドレスで直接指定される特別なページとして使うようになっています．その次の第１ページがスタック専用に使われるという設計です．これでも128回分の深さのサブルーチン呼び出しが可能だというわけです．スタック上にパラメータやローカルなデータなどは配置しにくいのですが，間接アドレッシングを上手につかえばデータ用のスタックやキューは，ソフトウェア的に簡単に実現可能ですから，そんなに無謀な設計ではありません．

命令セットの方は，さらに大胆に省略されています．たとえば加減算ですが，通常なら桁上がりが関係ないADD命令と，下位の桁からの桁上がりも同時に加えるADC命令の２つが用意されています．ところがMCS6502にはADD命令は存在しません．あるのはADC命令だけです．したがって，桁上がりを表すキャリーフラグを無視したADD命令相当のことを行わせようとするならば，キャリーフラグをクリアするCLC命令を前もって実行する必要があります．同様に，アキュムレータに対するインクリメントやデクリメント命令なども省略されています．さすがにインデックスレジスタに対して

のインクリメントは可能になっていますが，アキュムレータに関する操作などは必要最低限に近いほど減っています．

以上のような工夫により，MCS6502のチップ面積は小さくなっているといわれています．トランジスタ数が少なくチップ面積が小さいと，その分だけ歩止まりが上がりコストダウンにつながりますし，カスタムICのなかにCPUを組み込んでしまうのにも有利です．また，プロセスもN-MOSの一種の改良版を使っていることも重なって，高速のわりに低消費電力（MC6800や8080の半分以下）という特徴があります．

もうひとつ，10進演算が容易であるという点も特筆すべきものでしょう．コンピュータ内部で10進数表現を行う場合，BCD表示がしばしば使われます．BCD表示されている数値同士を2進数用の加減算命令で演算すると，桁上がりのしくみが違うため補正を行わないと正しいBCD表示の答えが得られません．一般的なCPUでは，この補正用の命令が用意されています（8086では，ADD命令に対してDAA命令が，SUB命令に対してDAS命令があります）．

ところがMCS6502には10進演算補正命令はありません．また10進演算用の特別な命令もありません．実はADC命令やSBC命令でBCD表示の数値を演算できるのです．えっ，たったいま2進数用の命令では正しい結果が得られないといったじゃないか，と思われた方もいらっしゃるでしょう．その矛盾を解く鍵はフラグの中にあります．レジスタセットの中のフラグ類がまとまっているコンデション・コード・レジスタにはDフラグというものがあります．8086のアセンブラをご存知の方は，ああ，ディレクション・フラグのことかと思われるかもしれませんが，こちらはデシマル・モード・フラグの略です．このフラグがリセットされていると，加減算の命令は通常の2進数演算を行います．一方，セットされていると，加減算の命令はBCD表示の10進数を演算するモードになって，自動的に補正が行われます．8080やMC6800には減算に対する10進補正命令はありませんでしたから，その点でも同世代のほかのCPUより一歩進んでいたといえるでしょう．

MC6800と類似だと思うと足をすくわれることもあります．アドレスを16bit長でメモリに記憶する場合，下位バイト・上位バイトの順に格納するのです．MC6800では上位バイト・下位バイトの順でしたから，逆になっています．8080の方に似ているのですね．

豊富なアドレッシングモード

独特の間接アドレッシング

MCS6502の最大の特徴は，そのアドレッシングモードです．

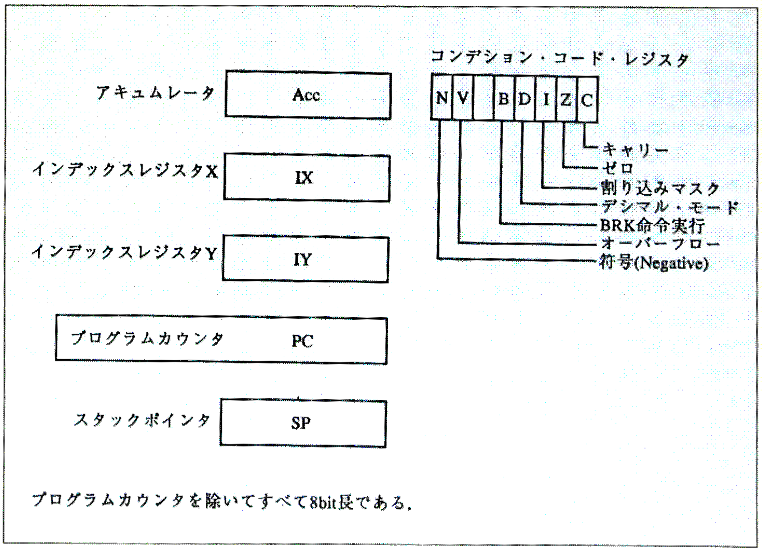
16bit長で直接アドレスを指定する絶対アドレッシングと，8bit長で0ページに含まれるメモリを直接指定する0ページアドレッシングは，MC6800と同じです．さらに絶対インデックスアドレッシングがMC6800のインデックスアドレッシングに相当しますが，これは前述のように16bit長のアドレスを8bit長のインデックスレジスタで修飾する形になります．この絶対インデックスアドレッシングは，基本的にIXレジスタとIYレジスタが両方とも利用できます．が，メモリのインクリメントやデクリメント，シフトなどの命令では，IXレジスタのみが利用可能です．以後さらにIXとIYレジスタの使い方が異なるということを説明しますが，このことからもわかるようにIXとIYレジスタには対称性はありません．

ここから先はMCS6502特有のアドレッシングモードになります．

IXレジスタには0ページ・インデックス・アドレッシングが使えます．8bit長のデータとIXレジスタの値を加算して，その下位8bitだけで0ページのアドレスを指定できます．

次は0ページ・インデックスの間接アドレッシングとでもいうモードです

図2　MCS6502のレジスタセット



(indexed indirect addressing). これもIXレジスタ特有のアドレッシングモードで，まず先に述べた0ページ・インデックス・アドレッシングと同じ操作をして，0ページのある特定のアドレスを決定します．次にそのアドレスから2Byteのデータを読み出して，読み出されたデータをアドレスと解釈して，実際の操作対象とします．0ページ・エリアにポインタを格納しておいて，ポインタを指定するのに0ページ・インデックス・アドレッシングを行うといえば，わかりやすいでしょうか．

もうひとつややこしいアドレッシングモードに，間接指定でオフセットを指定したインデックス・アドレッシング (indirect indexed addressing) があります．こちらはIYレジスタ特有のアドレッシングモードです．まず8bit長のデータで0ページのメモリ・アドレスを指定します．そこから2Byte長のデータを読み出して，その値にIYレジスタの内容を加算して，実際の操作対象のアドレスを求めます．

この2つのアドレッシングモードはわかりにくいと思いますので，何をやっているのかをC言語のプログラムにしてみたものを図3に付けておきます．複合的なポインタ操作の一種だと思えば間違いではないでしょう．

最後に普通の間接アドレッシングですが，これはJMP命令にのみ使えます．16bit長のデータで絶対アドレスを指定しますが，そこに格納されている2Byte長のデータをジャンプ先のアドレスと解釈します．こちらはほとんどおまけのようなものです．

いまでこそ間接アドレッシングを使えるマイクロプロセッサは多くなっていますが，まだ当時はミニコンピュータ・クラスでなければ用意されていませんでした．レジスタ類をCPU内に多数用意することができないことを補うためだとは思いますが，マイクロプロセッサに間接アドレッシングが使えるというのは画期的なことでした．

IXとIYレジスタを使った間接アドレッシングモードは，多数のポインタを実現するのに適しています．反面，16bit長のデータに対して1命令でインクリメントやデクリメントはできませんでしたから，ポインタが指すデータの長さが256Byteを超えると，ちょっと面倒になります．たいした手間ではないといえば，そうかもしれませんが．

以上のように，切り詰められた機能がある分だけ不便なところもありますし，MC6800と比べてワークエリアとして0ページを使いすぎて窮屈になりやすいところもあります．しかし，その風変わりなアドレッシングモードを使ってパズル的なプログラミングを行えば，想像以上の性能が引き出せたりするCPUでした．プログラミングしたとき，初心者と上級者の差が非常に大きくなるのです．そのような点が，初期のマニアに受けたのでしょうし，また実際にBASICを搭載したパーソナルコンピュータに使われたときにはベンチマーク・テストなどでも好成績を残したのでしょう．

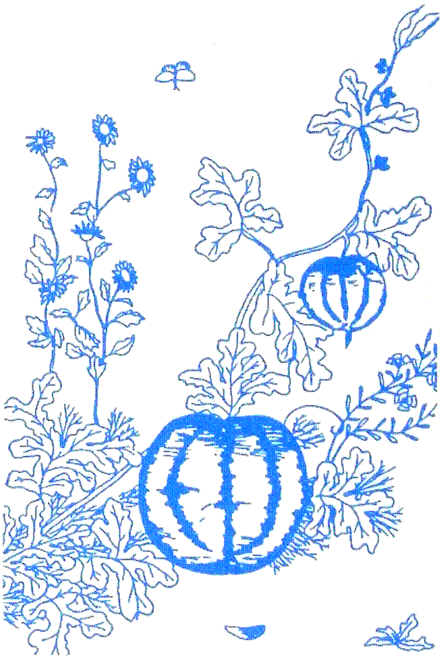
図3　MCS6502のアドレッシングモード

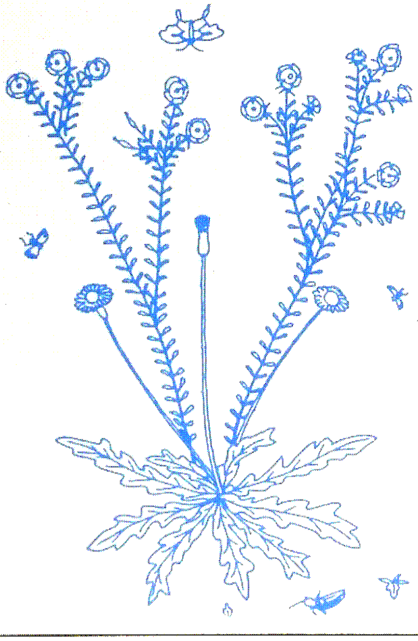
```
#define   BYTE char
#define   ADDR unsigned

BYTE Acc;           /* 当然アキュムレータ                    */
BYTE IX, IY;        /* それぞれIXとIYレジスタを表す           */
BYTE mem[65536L];   /* メモリ空間を表す                      */

/* 0ページ・インデックスの間接アドレッシングを利用して         */
/* Accに値をロードする関数                                    */
/*   IXレジスタだけが利用可能                                 */

void LDA_INDX(BYTE offset) {
ADDR a1, a2;
          /* まず0ページ範囲内のアドレスをインデックス修飾で計算して */
      a1 = (offset + IX) & 0xff;
          /* その場所からポインタをロードして、                     */
      a2 = ((ADDR)mem[a1 + 1] << 8) + mem[a1];
          /* そのポインタの指すアドレスからデータをロードする       */
      Acc = mem[a2];
}


/* 間接指定でオフセットを指定したインデックス・アドレッシングを用いて */
/* 同じことをする関数                                              */
/* こちらはIYレジスタだけが利用可能                                */

void LDA_INDY(BYTE zeropage_addr) {
ADDR a1, a2;
          /* まず0ページからいったんアドレスをロードして             */
      a1 = ((ADDR)mem[zeropage_addr + 1] << 8) + mem[zeropage_addr];
          /* その値とIYレジスタの内容を加えた場所から、値をロードする */
      a2 = a1 + IY;
      Acc = mem[a2];
}
```

プログラム遊工房

谷口典生

拾弐乃巻
スクリーンセーバ





今回は常駐プログラムの簡単な例として，スクリーンセーバ（screen saver，同じ画面が長時間表示されることによるモニタの焼き付きを防ぐための常駐ソフト）を作ってみました．ワンポイントテクニックでは，常駐プログラムを書くときのテクニックについて解説しています．

リスト1 SCSAVE.C

```c
/*
 * scsave.c            スクリーンセーバ
 */

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>

#ifndef __TURBOC__      /* Turbo C → MS-C(Quick C) 関数名変換 */
#define enable()        _enable()
#define inportb(x)      inp(x)
#define outportb(x,y)   outp(x,y)
#define setvect(x,y)    _dos_setvect(x,y)
#define getvect(x)      _dos_getvect(x)
#define keep(x,y)       _dos_keep(x,y)
#define freemem(x)      _dos_freemem(x)
#endif

#ifdef __TURBOC__
extern unsigned _heaplen = 512;         /* ヒープを縮小する */
extern unsigned _stklen = 512;          /* スタックを縮小する */
#define _chain_intr(vect) ¥
        asm     les     bx,vect ;¥
        asm     mov     ax,es ;¥
        asm     cli ;¥
        asm     mov     sp,bp ;¥
        asm     xchg    bx,[bp+14] ;¥
        asm     xchg    ax,[bp+16] ;¥
        asm     pop     bp ;¥
        asm     pop     di ;¥
        asm     pop     si ;¥
        asm     pop     ds ;¥
        asm     pop     es ;¥
        asm     pop     dx ;¥
        asm     pop     cx ;¥
        asm     retf
#endif

/* 割り込みハンドラの形式パラメータ */
#ifndef __TURBOC__
#define INT_PARAM       unsigned ES, unsigned DS, unsigned DI, unsigned SI, ¥
unsigned BP, unsigned SP, unsigned BX, unsigned DX, unsigned CX, unsigned AX
#else
#define INT_PARAM       unsigned BP, unsigned DI, unsigned SI, unsigned DS, ¥
unsigned ES, unsigned DX, unsigned CX, unsigned BX, unsigned AX
#endif

#define APP_NUMBER      0x9201          /* アプリケーション番号(int2fh用) */
#define INT2F_CHECK             1       /* int2fh用パラメータ値 */
#define INT2F_RELEASE           2
#define INT2F_EXIST             0       /* int2fh用戻り値 (!=APP_NUMBER) */
#define INT2F_DONE              0
#define INT2F_CANNOT_RELEASE    1

void (interrupt far *oldint2f)();       /* 元のint2fh割り込みルーチン */
void (interrupt far *oldint09)();       /* 元のint09h割り込みルーチン */
void (interrupt far *oldint29)();       /* 元のint29h割り込みルーチン */
void (interrupt far *oldvsync)();       /* 元のvsync割り込みルーチン */

int vsync_mask;         /* VSYNC割り込みのマスク状態 */
long counter=0;         /* vsync割り込みによるカウンタ */
long waittime;          /* 画面消去までの待ち時間 */
int shutflag=0;         /* 画面消去中フラグ */

void interrupt far vsync(INT_PARAM)             /* vsync 割り込みルーチン */
{
        if(!shutflag) {
                counter++;
                if(counter>waittime) {
                        shutflag = 1;
                        outportb(0x68,0x0e);    /* 画面表示を停止 */
                }
        }
        oldvsync();                     /* 元のVSYNC割り込みを呼び出す */
        outportb(0x64, 0);              /* VSYNC割り込みをリセット */
}

void interrupt far int09(INT_PARAM)             /* int09 割り込みルーチン */
{
        counter = 0;
        if(shutflag) {
                outportb(0x68,0x0f);            /* 画面表示を開始 */
                shutflag = 0;
        }
        _chain_intr(oldint09);          /* 元のint09hルーチンに処理を渡す */
}

void interrupt far int29(INT_PARAM)             /* int29 割り込みルーチン */
{
        counter = 0;
        if(shutflag) {
                outportb(0x68,0x0f);            /* 画面表示を開始 */
                shutflag = 0;
        }
        _chain_intr(oldint29);          /* 元のint29hルーチンに処理を渡す */
}

void interrupt far int2f(INT_PARAM)             /* int2fh ルーチン */
{
        unsigned far *p;

        if(AX!=APP_NUMBER) {                    /* 自分の番号でない */
                _chain_intr(oldint2f);          /* 元のint2fhルーチンへ */
        }

        if(BX==INT2F_CHECK) {                   /* 常駐のチェック? */
```

プログラムの使い方

このプログラムはPC-9801シリーズ用で，Turbo C, QuickCおよびMS-Cに対応しています．リスト1を打ち込み，コンパイルして実行ファイルを作ってください．

プログラムの実行は

scsave <待ち時間>

で行います．待ち時間は秒数で設定し，たとえば5分間なら5×60＝300秒で

scsave 300

と指定します(待ち時間のカウントはVSYNC割り込みで行っているので，正確な秒数というわけではありません)．実行すると，プログラムはメモリ上に常駐してキー入力と画面出力を監視し，最後にキーボードからの入力または画面への出力があってから指定した時間が経過すると，画面を消去します．なんらかのキー入力または画面出力が行われればすぐに元の画面が表示されます．

常駐を解除するには

scsave -r

を実行します．常駐したscsaveが占有していたメモリが解放され，動作を終了することができます．ただし，ほか

の常駐ソフトなどによってscsaveが使用している割り込みベクタが変更されていると，割り込みベクタを元に戻すことができないため，常駐を解除することができません．このときはあとから常駐させたプログラムを解除してから，再度scsaveの常駐解除を実行してください．

```
107:                 AX = INT2F_EXIST;
108:                 return;
109:         } else if(BX!=INT2F_RELEASE)
110:                 return;
111: 
112:         /* トラップしている割り込みベクタが変更されているか? */
113:         if(int2f != getvect(0x2f) || int09 != getvect(0x09) ||
114:            int29 != getvect(0x29) || vsync != getvect(0x0a)) {
115:                 AX = INT2F_CANNOT_RELEASE;
116:                 return;
117:         }
118: 
119:         p = (void far *)(((long)_psp<<16) + 0x2c);      /* 環境セグメント */
120:         freemem(_psp);                  /* プログラム領域を解放 */
121:         freemem(*p);                    /* 環境領域を解放 */
122:         setvect(0x2f, oldint2f);        /* 割り込みベクタを元に戻す */
123:         setvect(0x09, oldint09);        /* 割り込みベクタを元に戻す */
124:         setvect(0x29, oldint29);        /* 割り込みベクタを元に戻す */
125:         outportb(2, inportb(2) | vsync_mask);   /* VSYNC割り込みマスクを復帰 */
126:         setvect(0x0a, oldvsync);        /* 割り込みベクタを元に戻す */
127:         AX = INT2F_DONE;
128: }
129: 
130: void set_vector()                       /* 割り込みベクタの設定 */
131: {
132:         oldint2f = getvect(0x2f);       /* 元のint2fh割り込みベクタを保存 */
133:         setvect(0x2f, int2f);           /* int2f() をベクタに設定 */
134:         oldint09 = getvect(0x09);       /* 元のint09h割り込みベクタを保存 */
135:         setvect(0x09, int09);           /* int09() をベクタに設定 */
136:         oldint29 = getvect(0x29);       /* 元のint29h割り込みベクタを保存 */
137:         setvect(0x29, int29);           /* int29() をベクタに設定 */
138: 
139:         vsync_mask = inportb(2) & 4;    /* VSYNC割り込みのマスク状態を保存 */
140:         outportb(2, inportb(2) | 4);    /* VSYNC割り込みをマスク */
141:         oldvsync = getvect(0x0a);       /* 元の割り込みベクタを保存 */
142:         setvect(0x0a, vsync);           /* vsync() をベクタに設定 */
143:         outportb(2, inportb(2) & ~4);   /* マスクのクリア */
144:         outportb(0x64, 0);              /* VSYNC割り込みのリセット */
145:         enable();                       /* 割り込み許可フラグをセット */
146: }
147: 
148: void main(int ac, char *av[])
149: {
150:         union REGS regs;
151: #ifndef __TURBOC__                      /* ss,spの値を得る(MS-C) */
152:         unsigned _SS, _SP;
153:         _asm    mov     _SP,sp
154:         _asm    mov     _SS,ss
155: #endif
156: 
157:         if(ac!=2) {
158:                 puts("Usage : scsave {<待ち時間(秒数)>|-r}");
159:                 exit(0);
160:         }
161: 
162:         if(av[1][0]=='-' && (av[1][1]=='r' || av[1][1]=='R') && av[1][2]==0) {
163:                 regs.x.ax = APP_NUMBER;
164:                 regs.x.bx = INT2F_RELEASE;              /* 常駐解除 */
165:                 int86(0x2f, &regs, &regs);
166:                 if(regs.x.ax==INT2F_DONE) {
167:                         puts("常駐を解除しました");
168:                 } else if(regs.x.ax==INT2F_CANNOT_RELEASE) {
169:                         puts("割り込みベクタが変更されているので常駐解除できません");
170:                 } else {
171:                         puts("常駐していません");
172:                 }
173:                 exit(0);
174:         }
175: 
176:         waittime = atol(av[1])*56;              /* 1秒につきほぼ56カウント */
177:         if(waittime==0)
178:                 waittime = 10*60*56;            /* デフォルトは10分 */
179: 
180:         regs.x.ax = APP_NUMBER;
181:         regs.x.bx = INT2F_CHECK;                        /* 常駐チェック */
182:         int86(0x2f, &regs, &regs);
183:         if(regs.x.ax==INT2F_EXIST) {
184:                 puts("すでに常駐しています(常駐解除は scsave -r)");
185:                 exit(0);
186:         }
187: 
188:         puts("常駐します");
189:         set_vector();                           /* 割り込みベクタの設定 */
190:         keep(0, _SS + (_SP/16) + 1 - _psp);     /* 常駐終了 */
191: }
192: 
```

プログラムの説明

20-21行

Turbo Cで常駐サイズを小さくするため，ヒープ領域とスタック領域を縮小．

22-36行

Turbo C用にMS-Cの_chain_intr()関数に相当するものを定義．

40-46行

割り込みハンドラの形式パラメータ定義．割り込み処理関数では各レジスタをスタックに退避するコードが生成されます．形式的に関数のパラメータを定義することによってスタックに積まれたレジスタ値を参照，変更することができます．Turbo CとMS-Cではスタックに積まれる順序が異なるため，別々に定義しています．

48行

アプリケーション番号．int2fhでアプリケーションを判別するための番号で，プログラムによって違う値を使います．値について細かい決まりはないので適当な数字を定義しておきます．

49-53行

プログラム起動時の処理ルーチンからint2fh処理ルーチン（常駐部）に受け渡す機能番号と戻り値を定義．

❖vsync()

VSYNC割り込みルーチン．画面消去中でなければカウンタ値を増やし，設定した待ち時間を越えたら画面消去中フラグを立て，画面の表示を停止します．

❖int09()

int29()

キーボード割り込み（int09h）およびDOSの画面出力（int29h）割り込み処理ルーチン．それぞれの割り込みが発生した時点でカウンタ値を0に戻

し，もし画面消去中であれば画面を表示します．

❖int2f()

常駐プログラム通信用割り込み(int2fh) 処理ルーチン．

102-104行

AXレジスタの値が自分のプログラムで設定したアプリケーション番号と異なれば元のint2fhルーチンへ制御を移します．

106-110行

BXレジスタを使って指定された機能が常駐チェックならば，常駐中であることをAXレジスタに返します．常駐チェックでなく，常駐解除のコールでもなければ何もせず割り込み処理を終了します．

113-117行

プログラム内の処理ルーチンを設定した各割り込みベクタの値を確認し，1つでも変更されているものがあれば常駐解除できないことをAXレジスタに返します．

119-121行

常駐プログラムが使っている領域を解放します．PSP (プログラムセグメントプレフィクス) のオフセット2chには環境領域のセグメント値が格納されているので，これを利用して環境領域を解放します．

122-126行

トラップした割り込みベクタをすべて元に戻し，VSYNCの割り込みマスク状態を常駐前の状態に戻します．

127行

常駐解除したことをAXレジスタに返します．

❖set_vector()

プログラムでトラップする各割り込みベクタの値を退避し，プログラム内の処理ルーチンを各ベクタに設定します．VSYNC割り込みのマスク状態も退避します．

❖main()

151-155行

Turbo Cでは_SS, _SPでSS, SPレジスタの値を直接参照することができますが，MS-Cではこの機能がないので，インラインアセンブル命令を使ってSS, SPレジスタの値を参照します．

157-160行

引数なしで起動されたときは使い方を表示して終了します．

162-174行

スイッチ「-r」をつけて起動された場合，int2fhを利用して常駐解除を試みます．AXへの戻り値によって常駐解除ができたか否か，およびプログラムが常駐していないという結果を表示し，プログラムを終了します．

176-178行

起動時の引数から画面消去までの待ち時間を設定します．

180-186行

int2fhを利用してすでにプログラムが常駐しているかどうかをチェックし，常駐中ならその旨を表示し，プログラムを終了します．

188-190行

各割り込みベクタを設定し，プログラムを常駐終了します．

ワンポイントテクニック

常駐プログラムのまとめ

前回はcommand. comを子プロセスで起動する「常駐プログラムもどき」を作りましたが，今回は本当の常駐プログラム(以下TSR)です．

C言語でTSRを書くには，いくつか問題となる点があります．まず，プログラムを常駐終了するにはkeep() (MS-Cでは_dos_keep()) 関数を使いますが，引数として常駐するメモリサイズをパラグラフ単位で指定します．この長さを計算するためには，コンパイルされたプログラムが起動したときのメモリマップを知っておく必要があります．

大まかに説明すると，Turbo Cのスモールモデルではコードセグメント，データセグメント，ヒープ，スタックの順に並んでいて，MS-Cではコードセグメント，データセグメント，スタック，ヒープの順に並んでいます(ヒープは最初に割り当てられるもの)．

常駐サイズの算出にはいくつかの方法がありますが，今回のプログラムではヒープ領域は使わないので，単純にプログラムの先頭からスタックポインタまでを計算することにしました．プログラムの先頭はTurbo C, MS-Cともに_pspという予約変数によってPSPセグメントの値を知ることができます．スタックポインタの値は，Turbo Cでは_SS, _SPによって直接

レジスタの値を参照することができるのでこれを利用し，MS-Cではインラインアセンブラを使ってレジスタの値を変数_SS，_SPへ取り込むことにしました．指定はパラグラフ単位なので，セグメント値はそのまま，スタックポインタのオフセット値は16で割って1加え，最終的に(リスト190行目)「_SS ＋ (_SP/16) ＋ 1 － _psp」によって常駐するメモリサイズが得られます．

スタック領域も起動時にしか使わないので解放してしまってよいのですが，ここではスタックエリアを縮小することで対応します(Turbo Cではリスト21行目，MS-Cではリンク時に指定)．Turbo Cではヒープ領域がスタックの前にあるのでヒープは必ず縮小しておきます(リスト20行目)．

これで常駐するプログラムは書けますが，実際には常駐を解除する機能がないと不便です．また，スクリーンセーバが2重3重に常駐しても無意味ですから，すでに同じプログラムが常駐しているかどうかのチェックも必要です．常駐チェックには，特定のメモリに情報を記しておいたり，TSR自身がトラップする割り込みベクタの内容を調べたり，DOSが管理するMCB(メモリコントロールブロック)のリンク情報を追って調べたりといった方法がありますが，常駐プログラムとのインターフェースのために用意されているint2fh割り込みを利用するのがよい方法です．

int2fhを呼び出す側はAXレジスタにアプリケーション固有の番号を入れて呼び出し，int2fhをトラップしている処理ルーチン側はAXレジスタの値が自分のアプリケーション番号と一致するかどうかを調べます．一致した場合は常駐チェックや常駐解除のための情報の受け渡しをします．もちろんTSRへのパラメータの受け渡しも行うことができます．常駐の解除には退避したベクタ値やPSPセグメントの値を使用するので，int2fhの処理ルーチン側で割り込みベクタの復帰や領域解放を行ってしまったほうが楽です．

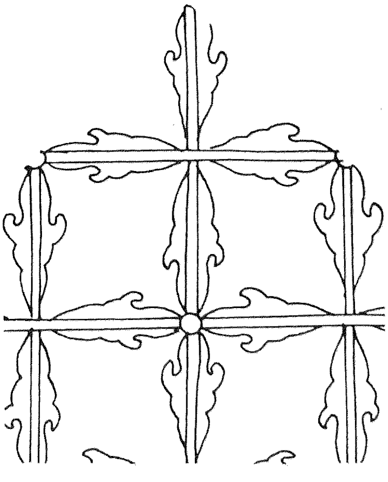
割り込みベクタを復帰するとき，トラップしたベクタの値が変更されているときは，ほかのプログラムがあとからベクタをさらにトラップしていることになるので，退避したベクタの値を書き戻すことはできません．このときは，割り込みベクタが変更されていることを表示し，常駐解除を中止したほうがよいでしょう．

割り込みルーチンのトラップについては前回も説明しましたが，ソフトウェア割り込みでは通常レジスタによるパラメータの受け渡しがあるので，元のルーチンを呼び出すときには，レジスタの値を最初に割り込みルーチンに入ってきたときの値に戻してやらなければいけません．さらに，Cの割り込み処理関数は最初にレジスタを退避しているので，元のルーチンを呼んで戻ってきたときのレジスタの値をそのままにして割り込み処理を終了するには，レジスタの値をスタックに書き戻してやることになります．スタックに積まれているレジスタの値は割り込み処理関数に形式パラメータを指定してやることで，通常の引数と同様にアクセスすることができます．レジスタの積まれる順序はコンパイラによって異なるため，リストではTurbo CとMS-Cそれぞれの形式パラメータを定義しています(リスト40-46行目)．

このようにソフトウェア割り込みをトラップするときのため，MS-Cには_chain_intr()という割り込みルーチンの引き継ぎをする関数があり，レジスタとスタックを割り込みが発生したときの状態に戻して，もとの割り込みルーチンへジャンプすることができます．この関数は有用ですが，Turbo Cにはこれに相当する関数がないので，インラインアセンブラを使って作ってみました(リスト22-36行目)．処理としては，最初にスタックに積まれるAXとBXレジスタの値と元の割り込みルーチンのベクタを入れ換え，retfでジャンプしてやることになります．この代替関数の利用に際しては1つ注意点があって，割り込み処理関数に引数もローカル変数もない場合に限ってはTurbo Cではspがbpに保存されない(mov bp, spのコードが生成されない)ため，正常に動作しません．リストのように必ず形式パラメータを指定してやれば問題ありません．なお，今回のプログラムをTurbo Cでコンパイルすると，使用していない形式パラメータについてうるさく警告メッセージが表示されますが，

```
#pragma warn -para
```

(パラメータを使用していない旨の警告メッセージを抑制する)という命令をリストの先頭につけ加えれば表示されなくなります．



イカすBASICプログラミング入門

第1回

構造化がカッコイイ!

酒井法雄

プログラミング環境は今や多極化しています．いまどきの言語を使いこなすには，非常にたくさんのことを覚えなくてはなりません．その中でも構造化プログラミングは共通のハードルです．この連載では，QuickBASICを使って構造化プログラミングの意義とコツを紹介し，カッコイイプログラミングを目指します．

パソコンでのプログラミングの歴史

プログラミングは，とても楽しくてワクワクするものです．一所懸命作ったプログラムが思ったとおりに動作したときは，何事にも代えられないほど嬉しい気持ちになるものです．それは，幾多ものハードルを越えてきたという自信にも繋がります．そんなことは本誌の読者の方々には説明するまでもないことでしょう．

しかし，そのハードルが年々高くなり，かつ多くなっていることにお気づきでしょうか？　ここで，ちょっと歴史を振り返ってみましょう．

パソコンという名前が定着する以前，個人が手に入れることができるマシンはマイコンと呼ばれる8ビット機でした．このころ，マイコンで何かをやろうと思ったら，BASICという言語は避けて通れないものでした．それは，OS自体がBASICであったからです．そのおかげで，巷には「BASIC教室」なるものが全盛を究めました．このころにマイコンを使ってプログラミングをしてきた人は，BASICやZ80のアセンブラでのプログラミングの経験があるはずです．しかし，それは必要から生まれたことであり，決して望んだからではなかったかもしれません．この時代には，BASICでの多少のプログラミングは必須であったのです．

パソコンの時代になって，MS-DOSが普及し，優れたアプリケーションソフトが増えてから，プログラミングの必要性はずっと少なくなりました．

時期を同じくして，プログラミング言語はCの時代へと移ってきました．Cはポータビリティに優れ，小さくて高速なオブジェクトを作ることができます．簡潔でフレキシブルな言語仕様はプログラマに喜ばれました．しかし，元BASICユーザーには思いもしないハードルが出現しました．1つは，コンパイラであったことです．BASICならばその場で「RUN」と打てば実行できたものが，エディタやコンパイラ，デバッガを行き来しないといけません．また，マシン語でしかできなかったような処理を簡単にできてしまうCは，使い方を間違えるとカンタンに暴走してしまいます．何よりも難しかったのは，プロシージャの概念と構造化を取り入れた言語仕様かもしれません．

このようにBASICよりハードルが高くなったCですが，妙なことに玄人好みするという噂のためか異常な人気になりました．巷にはCの入門書が溢れました．それはいまも続いています．しかし，「Cを使えます」というプログラマはたくさんいますが，Cの長所を

生かしたプログラミングができている人は，ほんのひと握りです．この現象は，「プロが使うものはいいものでカッコイイ」といった短絡的発想の産物かもしれません．

需要がありますから，メーカーも供給を考えないといけません．少しでもCを楽に使ってもらおうと，統合開発環境なるものが幅をきかせはじめました．しかし，これはプロやマニアにはあまりウケはよくなかったようです．やはり，いろいろなツールを使いこなしてこそプロの面目躍如ということなのでしょう．しかし，同時に高価だったCコンパイラも手軽に買える値段になってきました．

そうこうするうちに，今度はC++をはじめとするオブジェクト指向言語が流行りはじめました．ターゲットも，MS-DOSからMS-Windowsになってきました．ここで，またしてもたくさん覚えなくてはならないことが出てきました．オブジェクト指向の概念，Windowsについての知識，膨大な数のAPI関数，メッセージやプロパティ……．だんだん気が遠くなってきます．

この連載の趣旨

このように，プログラミングを取り巻く環境はつねに変化し続けています．そしてどんどん複雑になってきているのです．はじめてプログラミングをしようという人におすすめできる，カッコよくて流行の最先端のプログラミング言語は事実上ないのかもしれません．プログラミングには，どうしても「学習する」という側面があります．それは一朝一夕にできることではありません．あくまでもステップバイステップです．

そこで，まずは「近代的なプログラミングの基礎を学び」，「カッコイイプログラムを書けるようになろう」というのが本連載の趣旨です．具体的には，構造化を中心とした能率のよいスマートなプログラミングの方法を紹介します．

この連載で選んだ言語は，マイクロソフト社のQuickBASICです．QuickBASICは構造化文法やプロシージャの概念を取り入れたモダンなBASICです．インタープリタを内蔵していますから，すぐにプログラムを実行して経過や結果を見ることができますから，教材としては最適でしょう．

え？ Cじゃなきゃイヤだですって？ いきなりそんな高いハードルを飛び越えようとするのはやめておいた方がいいですよ．まずはQuickBASICで構造化と能率のよいプログラミングの手法を確立しておきましょう．Cやほかの言語に移行したときにも必ずやその経験は役立ちます．

見通しのよい簡潔なプログラム，これこそ通好みのカッコイイプログラムです．Cで書いてあればカッコイイなんていうのはお門違いです．Cで書かれていても，構造化されていない見通しの悪いプログラムではカッコワルイどころか醜悪です．

他人に「どう？このへんがちょっとカッコイイでしょ」などといえるプログラミングを目指しましょう．

構造化の話の前に

構造化とはよく聞く言葉です．しかし，その内容を把握して実際にプログラミングに生かしている人は少ないようです．難しそうな言葉だという先入観からかもしれません．また，他人のプログラムを見ればわかるが，自分ではなかなかそう書けないという人もいるでしょう．

これは，構造化という言葉が一人歩きしているからです．偉そうな言葉ですが，結局はプログラミングをラクにしよう，後からメンテナンスしやすくしようというようなことを実現するための一手法にすぎません．この本当の目的を忘れてしまっては元も子もないのです．

構造化プログラミングを実現するために，QuickBASICでは新しい文法やプロシージャの概念を言語仕様に取り入れています．これらを覚えないことには，構造化プログラミングはできません．しかし，本当に大切なのは「見通しのよいプログラミングを心がける」ことです．目的を忘れてた手段だけの方法論に陥ってしまってはいけません．

構造化とは

先ほども述べたように，構造化とは見通しのよいプログラミングのための手法です．

見通しの悪いプログラムは，バグが出やすく修正が難しいものです．また，後から仕様を変えようと中を見ると何をやっているのかわからなくなったりします．これでは，能率が悪くなってしまいます．

では，どうなっていれば，見通しのよいプログラムといえるのでしょう．それは，「何をやっているか」がすぐにわかるプログラムです．プログラムには目的があります．その目的を実現す

ワンラインNEWS
ミツミ電機はノート型パソコン用の入力デバイスとして小型トラックボール「TRB-S040I」を開発，来年2月から量産を開始する．

るためにいろいろな処理をします．しかし，自分のプログラムを後から見ると「ここは何をやりたいんだ？」と思うことがあります．この疑問にすぐに答えられるようにしたいわけです．

このために，コメントを書くという手法が使われてきました．しかし，実際にコードを書くよりもコメントを書くのに時間がかかったり，後から見てコメントだけではフォローできないこともあります．それは，おもに制御構造についての部分です．

リスト1a（sam_1_1a.bas）を見てください．これは，古い形式のBASICで書かれたプログラムです．何をやっているかわかりますか？　一見しただけではゴチャゴチャしていてわからないので，解読しないといけません．短いプログラムですから，流れを順に追って行きましょう．

100行目は，"OK? <y/else>"というプロンプトを出して，A$への入力待ちをしています．きっと，"yes"かそれ以外の文字列によって処理が変わってくるのだろうと推測できます．110行目で，A$が"Y"だったら200行目に飛びます．さて，ここでハタと困ります．ここからは200行目を見たらいいのか，それとも次の行を見たらいいのか……．とりあえず200行目からを見ると，"YES"という文字列を表示してBEEP音を出してプログラムが終了しています．では，A$が"Y"以外のときは……と120行目から見ると，"NO"という文字列を表示して，BEEP音を2回出して300行目に飛んでいます．300行目はプログラムの終了です．

入力された文字列が"Y"のときは"YES"と表示してBEEP音を出し，それ以外のときには"NO"と表示してBEEP音を2回出しているだけの，実に他愛のないプログラムなのですが，

```
100  INPUT  "OK?  <y/else>  :",  A$
110  IF  A$  =  "Y"  THEN  200
120  PRINT  "NO"
130  BEEP:  BEEP
140  GOTO  300
200  PRINT  "YES"
210  BEEP
300  END
```

リスト1a　sam_1_1a.bas

```
INPUT  "OK?  <y/else>  :",  a$
IF  a$  =  "Y"  THEN
     PRINT  "YES"
     BEEP
ELSE
     PRINT  "NO"
     BEEP:  BEEP
END  IF
END
```

リスト1b　sam1_1b.bas

どうもわかりにくいですね．では，いったいどこがわかりにくいのでしょうか？　使っている関数やステートメントはカンタンなものばかりです．コメントなんかつけたくもないくらい単純なプログラムです．

このプログラムは，「何をやりたいのか」がすぐにわからないのです．それは，やりたいことの単純さに比較して，プログラムの流れが複雑になっているからです．具体的には，行番号を使った2箇所のジャンプによって複雑にされています．IF…THEN構文の後の行番号指定やGOTO文によるジャンプは，処理の流れをつかみにくくする元です．やりたいことがもっと複雑になれば，プログラミングはその2乗倍くらい複雑になってしまいかねません．

このような制御の流れに関しては，コメントをいくら書いても解決にはならないでしょう．110行目に「A$がYだったら200行目に飛ぶ」などというコメントをつけても意味がないのです．なぜなら，「200行目に飛ぶ」ことは「やりたい処理」ではなくて「しかたないけどやらなくてはならない処理」で，その先のやりたい処理がすぐにはわからないからです．

このように，ジャンプはプログラミングを見通しの悪いものにする元凶です．しかし，ジャンプがなければ，かつての構造化言語仕様のないBASICなどではプログラミングが成立しなかったのです．そこで，フローチャートが活躍することになりました．フローチャートは，処理の流れを図で表すものです．このプログラムのフローチャートは，図1のように書くことができます．実はたいへん簡単なものであったことが，あらためてわかります．しかし，いちいちフローチャートとコードをつき合わせて見るというのも面倒です．実際には，フローチャートはお

おざっぱな処理については書くこともありますが，こんな細かい処理について書くことはまずないでしょう．

では，最初からこのフローチャートと同じようなイメージで，プログラムを書くことができればいいのではないかということになります．つまり，処理の流れをつかみやすいプログラムができればいいわけです．図2を見てください．これは，やりたいことをインデントで段付けし，わかりやすく羅列した文章です．この文章のイメージでプログラムを書くことができれば，見通しのよいプログラミングができそうです．

しかし，それにはプログラミング言語自体がそのような書き方を許さないといけません．QuickBASICやCは，このような書き方ができる構造化プログラミング言語です．たとえば，このプログラムをQuickBASICで書いてみると，リスト1b (sam1_1b. bas) のようになります．やりたいことがスグにわかりますね．

ここで注目したいのは，行番号がないことです．このおかげで，スッキリして見えます．構造化プログラミングでは，行を指定するジャンプを使う必要がほとんどなくなりますから，行番号は必要ありません．しかし，どうしてもジャンプが必要になるときにはラベルを使うこともできます．行番号がなくなったおかげで，後から一行だけ処理を入れたいときに「125 i=i+1」とか打ってから「RENUM」をする必要もなくなったわけです．

このように，構造化は見通しのよいプログラミングに極めて重要な役割を果たします．この他にも，QuickBASICではたくさんの構造化制御構文を備えています．それは，来月に詳しくご紹介します．

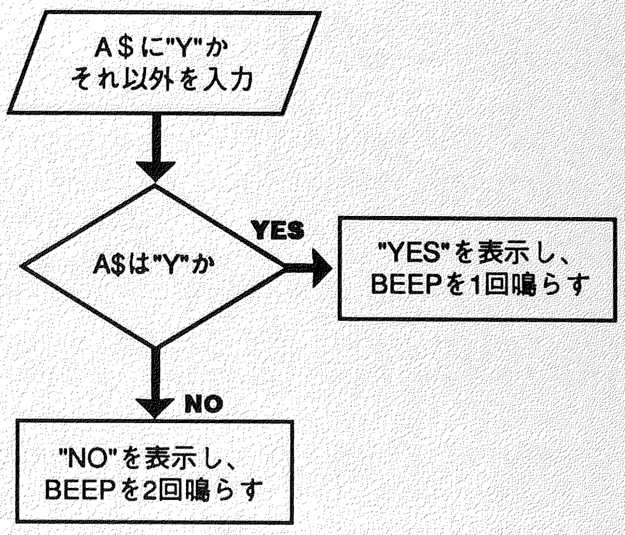
プロシージャ

構造化とともに，QuickBASICの言語仕様の目玉となっているのがプロシージャです．構造化文法を使えるだけではイマイチです．構造化とプロシージャをうまく使いこなせれば，「おっ，なかなかデキるな」と思われます．

なにやら横文字でめんどくさそうな名前ですが，やっていることは明快です．要は，関数やステートメントを自分で作ってしまおうということなのです．

先ほどのプログラムを考えてみましょう．入力した文字によって決まりきった処理をするというものですから，1つのプログラムの中でも何度も使うことになりそうなルーチンです．あちこちに何度も同じルーチンを書くのは，エディタのコピーとペーストの機能を使えばカンタンです．しかし，もし全部の"NO"を"No"に書き換えなくてはいけなくなったらどうしますか？　エディタの置換機能を使いますか？　間違って"NORIKO"を"NoRIKO"にしちゃったらどうしますか？　いや，それ以前に同じようなコードが

▼図1



▼図2

```
文字を入力する。
その文字が"Y"だったら、
        画面に"YES"と表示し
        ベルを1回鳴らす。
それ以外の文字だったら、
        画面に"NO"と表示し
        ベルを2回鳴らす。
おしまい。
```


ワンラインNEWS
東芝は記憶容量が130MBとこれまでで最大の2.5インチHDDを開発した．130MBのHDDを製品化したのは米クアンタム社に次いで2社目．

あちこちにあるのは，たいへんカッコワルイといわざるをえません．

そう，こんなときにはサブルーチンでした．しかし，このプログラムの中で使われているA$などという安直な変数名は，ほかでも使っているかもしれません（だいたいA$なんて変数名をつけること自体がカッコワルイですが）．変数の名前を気にしないで使えるサブルーチンがあれば……と思ったことはないでしょうか．それを実現するのがプロシージャです．

SUBプロシージャ

たとえば，YorElseというステートメントが最初からQuickBASICにあれば，こんなことで苦しまなくても済むわけです．じゃあ，ステートメントを作ってしまおうというわけです．

リスト2（sam_1_2.bas）を見てください．先ほどのプログラムの前後に「SUB YorElse」と「END SUB」という行が入っているだけです．この2つの文の間に入っているブロックの中では，ほかのルーチンと同じ変数名でも別の文字列として扱うことができるのです．メインルーチンからは，「YorElse」と1行書くだけでこのルーチンを呼び出すことができます．カンタンですね．これがSUBプロシージャです．

FUNCTIONプロシージャ

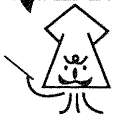
実際のプログラムでは，"Y"かそのほかの文字であったかの処理によって，その先の処理も変わってくるでしょう．SUBは呼び出すと特定の処理をしますが，SUBで行われた処理によって変数の内容が変わっても呼び出し側からはわかりません．

たとえば，YorElseという関数が最初からQuickBASICにあれば，こんなことで苦しまなくても済むわけです．じゃあ，関数を作ってしまおうというわけです．

リスト3（sam_1_3.bas）を見てください．先ほどのSUBがFUNCTIONになっているだけです．よく見ると，YorElseのあとに%が付いています．これはこの関数の戻り値が整数型であるということです．変数の後ろにつける%と同じです．このプログラムでは，"Y"を押されたときには1を，それ以外のときには0を返すようになっています．関数ですから，メインルーチンからは，PRINT文で表示するとか変数に代入するとかして呼び出します．どうですか，スマートでカッコ

```
DECLARE SUB YorElse ()
YorElse
END

SUB YorElse
     INPUT "OK? <y/else> :", a$
     IF a$ = "Y" THEN
          PRINT "YES"
          BEEP
     ELSE
          PRINT "NO"
          BEEP: BEEP
     END IF
END SUB
```

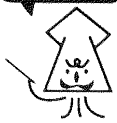
LIST 2



sam_1_2.bas

```
DECLARE FUNCTION YorElse% ()
PRINT YorElse%
END

FUNCTION YorElse%
     INPUT "OK? <y/else> :", a$
     IF a$ = "Y" THEN
          PRINT "YES"
          BEEP
          YorElse% = 1
     ELSE
          PRINT "NO"
          BEEP: BEEP
          YorElse% = 0
     END IF
END FUNCTION
```

LIST 3



sam_1_3.bas

イイでしょう．

このほかにも，プロシージャは引数をもつことができます．ある引数を元に，一定の計算をしてその結果を返す関数を作ることもできるわけです．また，プロシージャでメインルーチンの変数を共用することもできます．プロシージャからプロシージャを呼び出すこともできます．これらのプロシージャを構造化制御構文と組み合わせれば，いっそう見通しのよいプログラミングができるわけです．プロシージャ詳細については，来々月以降に解説します．

カッコイイ プログラミングのために

このほかにも，随所にカッコイイプログラミングのための技法があります．たとえば，変数名の付け方，ユーザー定義型と呼ばれるいろいろな型の変数を組み合わせて１つの変数型とするもの，名前付き定数の使用，インクルードファイルの使用，インデントの付け方，etc……．これらは来月以降紹介しますが，これ以降のサンプルプログラムでは勝手に使うこともあります．気にとめておいてください．

今月は，第１回ということもあり，カッコイイプログラミングのための主たる要素である構造化とプロシージャの簡単な紹介を通して，カッコよさ，見通しのよさについてのイメージをつかんでもらうことに専念しました．

「なあんだ．カンタンじゃないか」と思いましたか？　「こんなことは知ってるよ」と思いましたか？　前にも述べたように，カンタンだとか知っているとかは，カッコイイプログラミングのための本質ではありません．どういうモノがカッコイイかを知り，自分もそのカッコイイプログラミングを目指すことが大切なのです．

しかし，今回は「カッコイイ」というにはあまりにもカンタンな例だったかもしれません．では，最後にリスト４（sam_1_4.bas）に，先ほどのプログラムをちょっと拡張したプログラムを載せます．まだ紹介していない言語仕様も入っています．このプログラムのカッコよさを考えてみてください．

それでは，また来月お会いしましょう．

```
DECLARE FUNCTION YorElse% (ask AS STRING)
DECLARE FUNCTION GetOneChar$ ()
CONST YES = 1
CONST NO = 0

IF YorElse%("新型RX-7が好きですか") = YES THEN
    IF YorElse%("ふむふむ、それじゃ買いますか") = YES THEN
        PRINT "うわー。金持ちー!"
    ELSE
        PRINT "そうですよね。こうなったら旧型をチューンしてやる。"
        PRINT "しくしく。"
    END IF
ELSE
    IF YorElse%("それにしてもいい天気ですね") = YES THEN
        PRINT "どっか行きたいなぁ"
    ELSE
        PRINT "あ、それじゃ急ぐんで。"
    END IF
END IF
END

FUNCTION GetOneChar$
    DIM onechar AS STRING

    DO
        onechar = INKEY$
    LOOP WHILE onechar = ""
    GetOneChar$ = onechar
END FUNCTION

FUNCTION YorElse% (ask AS STRING)
    DIM yorn AS STRING * 1

    PRINT ask; "? <y/else> :";
    yorn = UCASE$(GetOneChar$)
    IF yorn = "Y" THEN
        PRINT "YES"
        YorElse% = YES
    ELSE
        PRINT "NO"
        YorElse% = NO
    END IF
END FUNCTION
```

LIST 4

sam_1_4.bas

ワンラインNEWS
韓国IBMは個人向けの16ビットパソコン「PS55-286モデル5510-ZL4」を発表した．従来，パソコン事業は業務向けだったが，個人需要の拡大をにらんで対応したもの．

Illustration ; Masami Nozue

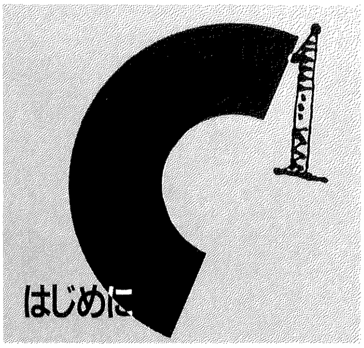
アルゴリズム篇 第1回

ソート(並べ替え)・その1

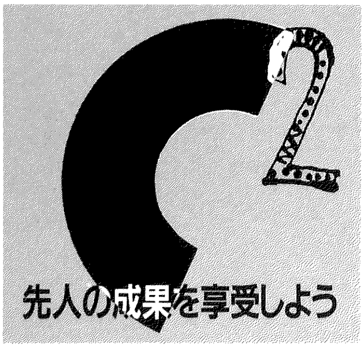
実践C塾

三代目塾頭
(株)ジャステックラボ
加野島英渡／山本博昭

今月から，先月まで「MS-DOSツール・ラボ」を担当していた加野島英渡氏を三代目の塾頭に迎えて，実践C塾アルゴリズム篇を開講する.「アルゴリズムを特定のプログラミング言語で表したものがプログラムである」というとおり，アルゴリズムはプログラミングの本質である，じっくりと学んでいただきたい.



今回から実践C塾の第3弾として，このコーナーを担当させていただくことになった．処理系の話，基礎テクニック篇と続いてきた当塾歴代塾頭の方々の足跡を汚さぬようにと緊張の面もちでキーボードに向かっている．さて，第3弾は「アルゴリズム篇」と銘打ってプログラムの基本的アルゴリズムに触れつつ，それらを具象化したツールを書いていくという進行になる．新連載ということで，いたらない面もあると思うが，いままでのアルゴリズム解説になかった新鮮な切り口をお見せできるのではないかと勝手に思い込んでいる．なにとぞよろしくご愛読のほど願う次第である．また，弊社も10月から人手が若干増加して，まがりなりにも会社としての体裁になってきたということで，今回の連載は共同執筆という形をとり，全社一丸となって取り組んでいくつもりである．そちらも合わせてよろしくお願いしたい．



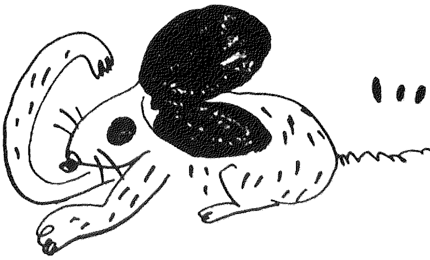
何かの漫画だったと思うが，学者連が遺跡の発掘現場で「おお，この石垣の組み方は 力学的にみても完璧だ．古代の技術はここまできていたのか」などと感動しているところへ，アルバイトの学生が「たまたま完璧だったのだけ残ってるんじゃないすか？」などと身もふたもないいい方をしていた場面があった．

ツールなどを何本も書いていると，しばしばこのような場面に出くわすことがある．あるデータ処理をするルーチンを書いていて，「以前似たような処理があったはずだが……」などと古いソースを引っ張りだしてきて，それに改良を加えて使っていく．もっと効率のよい処理方法を思いついて改造を加えたりして何回か使い回していると，次第にそのロジックは磨かれていって，あとで眺めてみるとみごとに理にかなっていたりする．もちろん，その裏には，1度限りで使い捨てられていったロジックたちが死屍累々となっているわけで「たまたまよかったのだけ残っている」という状態である．

これは個人にかぎったことではなく，コンピュータプログラミングの歴史上，「こうしたほうがいい」という改良は果てることなく続いており，その蓄積の1つの結晶ともいうべきものが各種のプログラミングアルゴリズムである．アルゴリズムの洗練は，「よりたくさんの人が必要としている」ところから発生してくる．「98ノートで親指シフト入力を使いたいが，{}の記号を楽に入力するためにはどのようなキー操作がよいか」などという特殊なニーズには，百花繚乱のアルゴリズムが競い合うということはないだろう．しかし，たとえば基本的なデータの整列や分類，検索といった「よく使う」ロジックには，コンピュータプログラムの創世期から数知れないプログラマたちの意趣をこらした手法が蓄積されている．

「アルゴリズム」というものは情報処理などの専門分野の教育機関やプログラマの新人研修などでは教えていただけるものらしいが（不幸なことに筆者はそういった学校や研修の経験があまりなかった），いざ実戦の場に出てみるとなかなか系統だった学習やその応用をするのはむずかしいものである．とくにツールなどはその場でバタバタと書いて「結果オーライ」になるケースが多いため，なぜこうするかという理論は忘れさられることがよくある（おおいに自戒の念をこめて書いている）．

「気がついたら理にかなっていた」というスキルも必要だろうが，ごくごく基本的なアルゴリズムを知っておくことによって，実戦のプログラミングの手助けになることが必ずあるだろう．アルゴリズムは決して机上の理論だけではなく，プログラミングの現場で生きるものであるからである．それにもう1つアルゴリズムを身に付けると意外な余録があるかもしれない．こうした専門用語は一般社会にはあまり流通していない用語が多いので，コンピュータを前にして「ここのロジックは×××法でいったほうがいいな……」などとつぶやいてみせたり，酒の席のちょっとした会話にさりげなく使ってみせたりするとソフトウェアエンジニアとしての株があがる可能性があるということである．ただし，周囲がエキスパート集団で，こうした用語が基礎の基礎である場合には何の効果もないことは確かだが…….



ワンラインNEWS
日本電気はソフトウェアを迅速に開発するためのソフトウェア自動合成エキスパートシステム「SOFTEX」を開発した．当面は社内での利用を進め，将来外販も検討する．

C3 この連載の進め方

そういうわけで，これから数回にわたって，いくつかのアルゴリズムの基本的な考え方とそれを使った実際のプログラムを紹介していく．まずは，アルゴリズムがいちばん豊富と思われる「データ構造とその処理」の周辺から話をはじめよう．

プログラムを作る上でも，「データ構造とその処理」は扱う頻度の高いものであろう．たとえば，あるデータ群の中から必要なデータを検索する必要が出た場合には，検索のロジックはもちろん，検索の効率アップのために前もって並べ替えを行うとか，場合によってはそのデータ群の構造自体の見なおしなども考えられる．さらさらと書いたが，いま出てきた処理はそのまま

- **サーチ**
- **ソート**
- **ツリー構造**

といったアルゴリズムの代表的なジャンルにあてはまる．これらのアルゴリズムを果たしてどういう形で実際のプログラムの中で使っていくのか，これから始まる連載はこのことを中心に進行していく．

ただ，実際のプログラミングとはいっても，断片的なロジックを書き連ねていくのでは読者も筆者も退屈してしまうのではないかと思い，何か1本のツールをテーマにして，新しく出てきたアルゴリズムを使ってそのツールを次々とバージョンアップしていくということにした．次章でテーマとなるツールの基本的な仕様をまとめる．

C4 テーマツールの仕様について

これからみていくさまざまなアルゴリズムを実験するために，1つのツールを作り，毎月出てくる新しいアルゴリズムを使って改良していくということになった．そうはいうものの，実験のためだけにプログラムを作るというのもつまらないし，後々の応用なども考えて，「ファイルの中の単語の出現頻度を調べる」というツールを作っていくことにした．基本的な仕様は，次のとおりである．

- 指定されたテキストファイルを読み込んで単語ごとに切り出し，その出現回数をカウントする．
- 全単語をカウントし終ったら，結果をABC順に表示する．
- ファイルは複数指定が可能なものとし，そのすべてのファイルの中の全単語をリストにする．
- プログラム名は「WCOUNT.EXE」とする．
- 起動方法は図1の通り．

C5 今回の重点項目

ここは本来，その回のテーマツールの改良箇所を書くべきところなのだが，今回は初回ということで，改良どころではなくテーマツールのプログラム作成をはじめから行わなくてはならない．基本仕様は前章でまとめた通りなので，ここではその仕様をもとにして実際のプログラムの動作を組み立ててみる．複数のファイルを処理していくということで，メインの処理は，1ファイル内の全単語を切り出して出現回数をカウントするというのを起動オプションで指定されたファイル名があるかぎり繰り返すループとなる．このループを抜けたところで，結果を表示して処理を終了する．

つぎに1ファイルごとの処理だが，おもな機能は「ファイルからデータを読み取って，単語を切り出す」「その単語をリストに登録し，出現回数をカウントする」という2つになる．切り出した単語をリストにするために何らかのデータ構造が必要になるが，とりあえずは，カウンタと文字列へのポインタが要素となる構造体を配列にすることで実現できそうである．

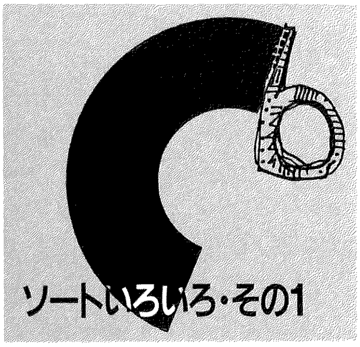
問題は，最後にその結果をABC順に表示するという処理である．ファイルを順に読んで単語を切り出し，そのリ

図1

```
A>WCOUNT <ファイル名> <ファイル名> <ファイル名> ・・・・
```

マイクロソフトはMS-Windows 3.0に対応した表計算ソフトウェア「MS-Excel」のモニターキャンペーンを92年1月31日まで行う．

ストを作成していったとしたら，その単語リストはABC順には並んでいない．これを表示のときに並べ替える処理が必要になってくる．実はCコンパイラの処理系には並べ替えのための関数が用意されているのだが，説明の都合上，そこは見なかったことにして，今回はこの「並べ替え」からプログラムをはじめることにする．



むりやり話をもってきたような感がなきにしもあらずだが，「アルゴリズム」といえばこのジャンルが反射的に浮かんでくるという噂も高い，「ソート」の説明をはじめてしまおう．コンピュータに何かデータ処理をやらせる場合，必ずといっていいほど「並べ替え」が必要になる．データベースなどの本格的なデータ処理まで考えなくても，ファイル名なり単語のリファレンスなりといった「一覧表」を作成する処理には，ABC順や日付の古い(または新しい)順などの並べ替えをすることになるだろう．このように必要性の高いものであるから，ソートは「アルゴリズムの宝庫」といっても過言ではないほどたくさんの手法が編み出されてきた．そのそれぞれに長所・短所があり，処理すべきデータが1バイトデータだったり，可変長の文字列だったり，またデータの状態や数によって適合する手法は変わってくる．「どんなデータがどんな状態できても絶対にこれがいちばん早い」という究極の並べ替えはまだこの世に存在しないのだ．

では，その「ソート」のアルゴリズムをいくつか紹介していこう．

選択ソート

数字の書かれているカードを適当にばらまいて，小さな子供に「小さい順に並べなさい」といってみる．その子がおとなしくこちらのいうことを聞いた場合，どのようにして並べていくだろう．いきなり高度なアルゴリズムを使うようなIQの高い子供もいるかもしれないが，一般庶民の子供の多くは，以下のような方法を取るのではないかと思われる(筆者の家に3歳児がいるのだが，やはりその庶民的な方法をとったのを実際に確認している)．

ばらばらになったカードの中から「1」のカードを探して，ほかの場所へ置く．次は「2」を探して，「1」の横に置く．これをばらばらになったカードの集まりがなくなるまで繰り返す．

3歳児でも思いつくのだから，これがもっとも原始的な「ソート」の手法と思われる．これをいい大人が整理してみよう．あるデータ列の中をすべてチェックしていって，その中のいちばん小さな値を探す．これがカードであればどこか別の場所に置いておくこともできるのだが，コンピュータの場合，うっかり変なところへ置くと後々問題が出てくるので，ここは「データ列の先頭と交換する」ということにする．交換したデータはデータ列から切り離して考えるようにする．つまり，1回のチェックでデータ列が1つずつ少なくなっていくことになる．残ったデータ列を対象に前回と同じことを繰り返す．データ列がなくなったら並べ替えが終わっていることになる．この手法はデータ列の中から，最小の値を逐一選択していくので「選択ソート」と呼ばれている．図2が選択ソートの方法である．

(1)がソート開始時のデータ列である．5個のデータがランダムに並んでいる．1回目の処理ではこの5つの中から最小の値を探す．結果として右端にある3が最小ということになるので，このデータと左端の5を交換する．(2)の2回目の処理開始のときには，左端の3はデータ列からはずれているので，残り4個のデータの中からさきほどと同じように最小のデータを探して，その時点でのデータ列左端のデータと交換する．この要領でデータ列を1つずつ減らしながら並べ替えを行う．最後は（5）のようにデータ列が1つになるので，交換は発生しなくなるため，ここで処理は終了ということになる．

「選択ソート」は，感覚的にいちばんわかりやすい手法ではあるが，効率はあまりよいものではない．とにかくすべてのデータを何度も何度も見ていかなくてはならないのだ．これが10個くらいのカードを人間が並べるのなら，「一目見たらいちばん小さいのがぱっとわかる」という技も使えるが，3歳児にもできる「ぱっと見てわかる」が現在のコンピュータではできないので，全部律義に比較していくわけである．

極端な話をするが，最初からきちんと並んでいるデータ列をソートさせてみても，選択ソート法ではすべてのデータの比較をいちいち行って，処理の結果，実行前と寸分違わぬデータとなるのだ（1度も入れ替えが発生しないのだから，データ列が変わっていない

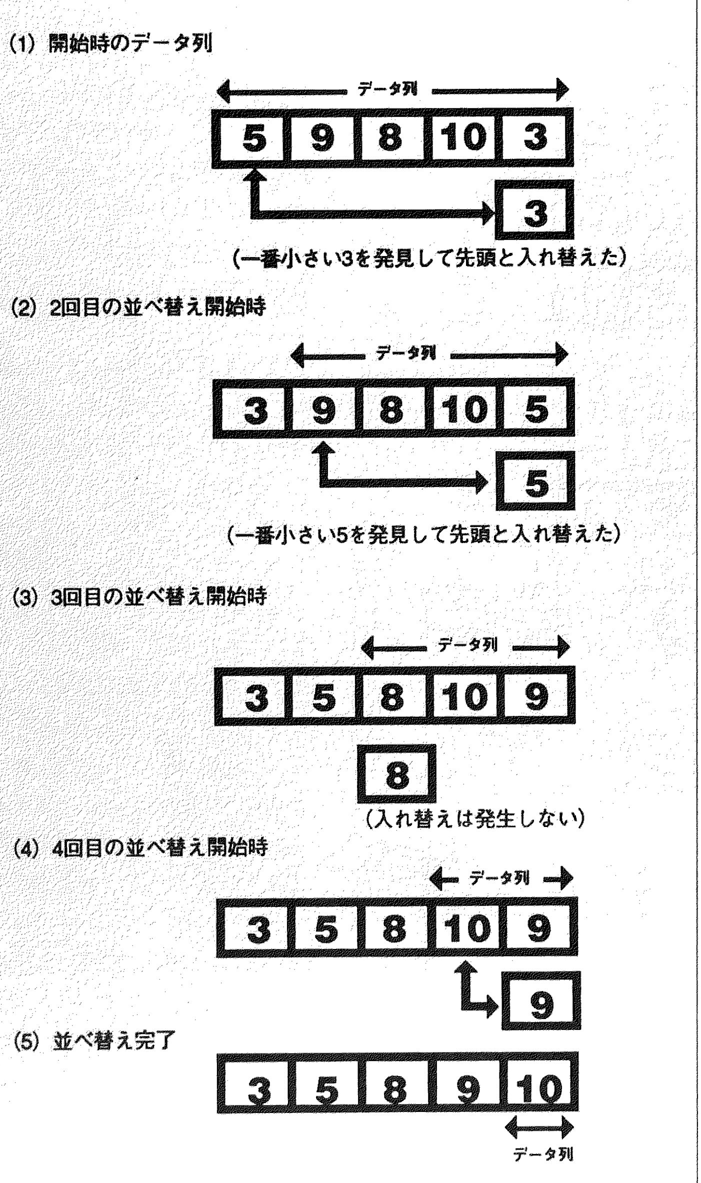
ワンラインNEWS
日本IBMはパソコンを使って点字による単語検索が可能な英和辞典「点字英和辞典システム」を開発した．視覚障害をもつ大学生や大学受験生に無償で貸し出す．

のは当然だが）．悪口ばかり書いてしまったようなので，少しだけ弁護すると，どんなデータのときも同じように律義に処理をしていくので，この手法はすばらしく早く終わることもないかわりに，思ったより時間がかかることもない．大勝ちしないが大崩れもしない中堅どころといった手法である．

バブルソート

その名前を聞いて，「バグるソート」といういい回しがすぐに脳裏に浮かんでしまうが，「バブルソート」はロジックの明快な書きやすいアルゴリズムである．その基本的な発想は「隣どうしを比較して，順序が逆だったら入れ替えていく」というものである．これを総当たり制でやっていく．比較の結果，大きいほうが「勝ち」として話を進めていくと，全データの1回目の対戦が終わったときには，すべての対戦に全勝したデータが右端に残っているはずである．第2回戦はこの最強データを除いた残りのデータで行われる．最終戦を終えたときには「1回も勝てなかった最弱データ」が左端に残されて，熱戦の幕は降ろされる．

図2　選択ソートの方法



ではこれを具体的なデータで見ていくことにする．図3の（1）がもともとのデータ列である．比較は左端の5とその1つ右側の10からはじまる．これは10の「勝ち」なので入れ替えは発生しない．次の対戦は場所を1つ右へ移動して10と8の比較となる．これは10のほうが大きいので入れ替えが生じる．そしてまた1つ右へ移動して比較するわけだが，このときの左側のデータ位置には，さきほど入れ替えた10が入っているので，10と9の比較となる．当然，ここでも10は勝利をおさめることになるのである．こうして，比較が右端まで達したとき，10はすべての比較に勝って右端の位置に収まっている．第2回戦は（2）の時点で位置の確定した10を除いた4つのデータではじまる．つまり，比較する範囲が右端から1つずつ狭くなっていくわけだ．（3），（4）と比較はすすみ，（5）までくると，位置の確定していないデータは1度も比較で勝利をおさめることのできなかった左端の3だけとなる．これではもはや比較はできないので処理はここで終わりとなる．

さて，ここで右側のページが上になるように「プログラマーズ・ページ」を90度まわして，いまの図をもう1度見てほしい．第1回戦では10が，第2回戦では9が，というようにそのときの「勝者」が上のほうに浮かび上がってきてその位置を確定している．この

「ぷくぷく」と浮かび上がってくる様子が「バブルソート」の名前の由来である．近頃は「バブルははじける」ことになっているようだが，アルゴリズムの世界での「バブル」は着実に並べ替えを実行してくれる．

では実際にこのソート手法をCでコーディングしてみる．リスト２の「BSORT.C」をご覧いただきたい．このプログラムは，配列として定義された７つの数値をバブルソートを用いてソートしながらその途中経過を表示するものである．このプログラムは，Borland C++でコーディングしてあるが，リスト１のMAKEファイルを修正するだけで，ほとんどのCコンパイラでそのままコンパイルできるはずである．起動は単純に

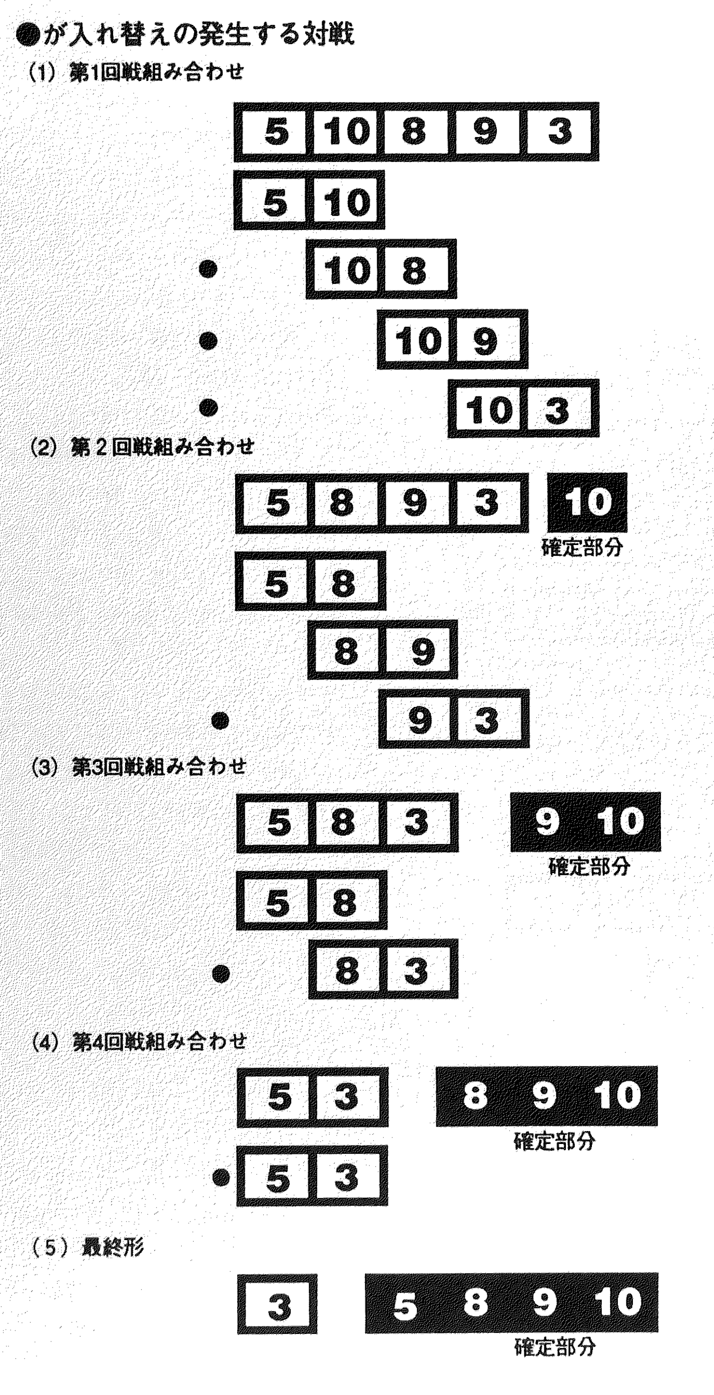
```
A>BSORT
```

とする．最初の配列の表示からはじまって，ソートの途中経過を逐次表示しながら処理を進める．最後の表示行がソートの結果である．

実際のバブルソートのロジックは，途中経過の表示部分を除けば，図４にあるように非常にシンプルである．全体は２重ループになっていて，外側のループは配列の個数回だけまわる．内側のループは配列の個数回からはじまって，処理のたびに１つずつ処理回数が減っていく．比較はつねに左端から処理をはじめて，確定した右側のデータを除いた範囲でソートを行っていくからである．この範囲内で次々と１つのデータとその右側を比較して，逆順になっていればswap_data関数を使ってこの２つを入れ替える．実行経過が表示されるので，major_countが-1されるたびに，データ列の中でいちばん大きな値が右側に「浮かび上がってくる」のを確認してほしい．

図５がBSORT.EXEの実行画面である．ソートすべき配列の内容は，プログラムの冒頭に要素数とともにコーディングしてあるので，これをいろいろと変えてソートの様子を表示させてみるとこの手法の特徴がわかるのではないかと思う．

図３　バブルソートの方法



ワンラインNEWS
日立マクセルは3.5インチ光磁気ディスクを開発，サンプル出荷を開始した．磁性膜を２層構造にし，従来の半分の磁場で信号を記録・消去できる．

リスト1 BSORT.MAK

```
.AUTODEPEND

#                   *Translator Definitions*
CC = bcc +BSORT.CFG
TASM = TASM
TLINK = tlink

#                   *Implicit Rules*
.c.obj:
  $(CC) -c {$< }

.cpp.obj:
  $(CC) -c {$< }

#                   *List Macros*

EXE_dependencies =  ¥
  bsort.obj

#                   *Explicit Rules*
bsort.exe: bsort.cfg $(EXE_dependencies)
  $(TLINK) /x/n/c/P-/LA:¥BC¥LIB @&&|
c0s.obj+
bsort.obj
bsort
                    # no map file
emu.lib+
maths.lib+
cs.lib
|

#                   *Individual File Dependencies*
bsort.obj: bsort.c

#                   *Compiler Configuration File*
bsort.cfg: bsort.mak
  copy &&|
-v
-vi
-IA:¥BC¥INCLUDE
-LA:¥BC¥LIB
-P-.C
| bsort.cfg
```

リスト2 BSORT.C

```
/* ==============================================================

    BSORT.C         Copyright 1991 by JUSTECH-LABO. (H.kanoshima)

============================================================== */
#define DSIZE   7
#define DLEFT   0
#define DRIGHT  (DLEFT + DSIZE - 1)

int     base_data[DSIZE] = {7 , 6 , 5 , 4 , 3 , 2 , 1};

/* ============================================================== */

void    dump_data(char *guide)
{
int     count = DSIZE;
int     i = DLEFT;

    printf(guide);

    while(count--)
        printf("%4d",base_data[i++]);

    puts("");
}

/* ============================================================== */

void    swap_data(int x , int y)
{
int     tmp;

    tmp             = base_data[x];
    base_data[x]    = base_data[y];
    base_data[y] = tmp;
}

/* ============================================================== */

void    bsort(initial_left , initial_right)
{
int     index , count;
int     major_count = initial_right - initial_left;

    dump_data("(--start) ");
    while(major_count)
    {
        index = initial_left;
        count = major_count;

        while(count)
        {
            if(base_data[index] > base_data[index + 1])
                swap_data(index , index + 1);
            index++;
            count--;

            dump_data("(  minor) ");
        }
        dump_data("( -major) ");
        major_count--;
    }
}

/* ============================================================== */

void    main(void)
{
    bsort(DLEFT , DRIGHT);
}
```

図4 バブルソートの実際のコーディング

```
:       while(major_count)
```
外側のループ（配列要素数回まわる）

```
:       {
:               index = initial_left;
:               count = major_count;
```
内側のループ開始時に，左側のデータポインタとループカウンタを初期化する．

```
:               while(count)
```
内側のループ

```
:               {
:                       if(base_data[index] > base_data[index + 1])
:                               swap_data(index , index + 1);
```
逆順になっていれば，データを入れ替える．

```
:                       index++;
```
比較位置を1つ左へずらす．

```
:                       count--;
```
カウンタを－1する．

```
:               }
:               major_count--;
:       }
```

図5 BSORT.EXE実行画面

```
A>BSORT
(--start)   7   6   5   4   3   2   1
(  minor)   6   7   5   4   3   2   1
(  minor)   6   5   7   4   3   2   1
(  minor)   6   5   4   7   3   2   1
(  minor)   6   5   4   3   7   2   1
(  minor)   6   5   4   3   2   7   1
(  minor)   6   5   4   3   2   1   7
( -major)   6   5   4   3   2   1   7
(  minor)   5   6   4   3   2   1   7
(  minor)   5   4   6   3   2   1   7
(  minor)   5   4   3   6   2   1   7
(  minor)   5   4   3   2   6   1   7
(  minor)   5   4   3   2   1   6   7
( -major)   5   4   3   2   1   6   7
(  minor)   4   5   3   2   1   6   7
(  minor)   4   3   5   2   1   6   7
(  minor)   4   3   2   5   1   6   7
(  minor)   4   3   2   1   5   6   7
( -major)   4   3   2   1   5   6   7
(  minor)   3   4   2   1   5   6   7
(  minor)   3   2   4   1   5   6   7
(  minor)   3   2   1   4   5   6   7
( -major)   3   2   1   4   5   6   7
(  minor)   2   3   1   4   5   6   7
(  minor)   2   1   3   4   5   6   7
( -major)   2   1   3   4   5   6   7
(  minor)   1   2   3   4   5   6   7
( -major)   1   2   3   4   5   6   7
```

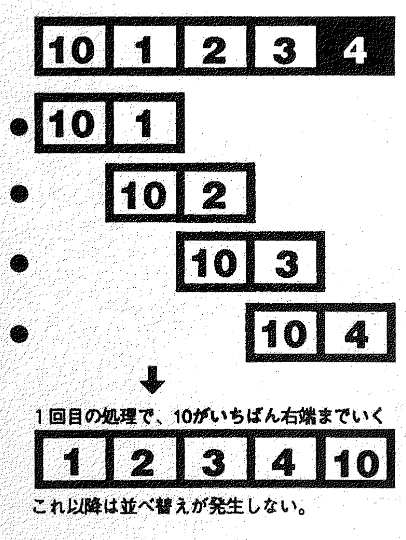
ワンラインNEWS
YHPはWindows 3.0にはじめて対応したリアルタイムＣデバッガー「HP64700」を開発した．日本電気のPC-9800シリーズに対応．

改良型バブルソート

「バブルソート」は，図6のようなデータを想定すると，もっと効率のよい改造ができる．第1回戦で10が右端まで「浮かび上がって」くるのだが，10の位置が確定してしまうと，もうどの対戦でも並べ替えは発生しないことになる．通常はデータが5つのときは4回戦まで(つまり，データ数－1回戦)処理をしないといけないことになるのだが，途中ですべてのデータがきれいに並んでしまった場合にはその時点で打ちきることができるのだ．

プログラムで途中終了できるかどうかを判定するには，1回の処理の中で1度も入れ替えが発生しなければ（今の例の場合は，2回戦において1度も入れ替えが発生しなくなる）打ちきりというロジックにすればよい．たとえば，処理の前にフラグをクリアしておき，入れ替え処理でフラグをセットするということにしておくと，1回分の処理が終わってフラグがたっていなければ入れ替えが発生しなかったと判定できる．こういった手法を「フラグ付きバブルソート」などと呼んでいる．

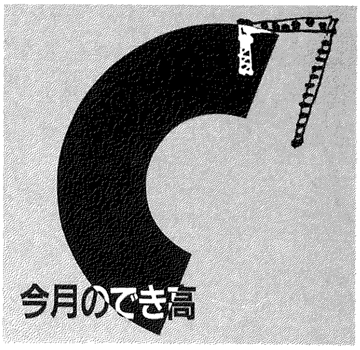
図6　バブルソートの端的な例



実際のバブルソートプログラムを「フラグ付き」にするのは，非常に簡単なことではある．図7が，さきほどの「BSORT.C」のソート部分をフラグ付きにしてみたものであるが，たった3行を追加するだけである．1回の処理で1度も入れ替えが発生しなかったということは，プログラムの内側のループで1度もswap_dataが実行されなかったことを意味しているので，swap_flagというフラグを用意して，swap_dataを実行したときにTRUEにする．これがデータの入れ替えがあったという状態である．内側のループを抜けてきてこの変数がTRUEになっていなければ，1度も入れ替えが発生しなかったわけだから，ソートはここで終了してよい．先に書いたようにこの改造を行うとあらかじめ整列しているデータだった場合には，ループ回数が少なくてすむ．ただしどんな場合でも絶対に早くなるとは限らない．フラグの初期化，設定，判定のロジックが余分に追加されるわけであるから，最後の1回まで入れ替えが必要だったらかえって遅くなってしまうからである．

図7　バブルソートのロジックをフラグ付きに改造した例

```
 :      while(major_count)
 :      {
 :              index = initial_left;
 :              count = major_count;
>:              swap_flag = FALSE ;
入れ替えフラグをクリアしておく

 :              while(count)
 :              {
 :                      if(base_data[index] > base_data[index + 1]) {
 :                              swap_data(index , index + 1);
>:                              swap_flag = TRUE ;
入れ替え処理を実行したときにフラグをたてる

 :                      }
 :                      index++;
 :                      count--;
 :              }
>:              if (swap_flag == FALSE) return ;
フラグが立っていなければ，1度も入れ替えがおきなかったと判定できる．

 :              major_count--;
 :      }
```



実行ファイルの作成方法

この連載でのプログラムの作成には，Borland社のBorland C++ Ver.2.0を使用している．掲載したソ

ワンラインNEWS
アイ・オー・データ機器，関西電機は共同でプリンタバッファを開発した．プリンタバッファとしてははじめてDMA方式を採用し，処理速度を上げた．

ースのMAKEファイルをリスト3に示す．なお，このMAKEファイルは，Borland C++の環境版のプロジェクトファイルを「PRJ2MAK」というツールで作成したものである．

```
A>MAKE -fWCOUNT.MAK
```

で，実行ファイルWCOUNT.EXEができあがる．

使い方

テーマツールの基本仕様のとおり，起動オプションとして処理対象のファイル名を指定する．ファイルは空白で

リスト3　WCOUNT.MAK

```
.AUTODEPEND

#               *Translator Definitions*
CC = bcc +WCOUNT.CFG
TASM = TASM
TLINK = tlink

#               *Implicit Rules*
.c.obj:
  $(CC) -c {$< }

.cpp.obj:
  $(CC) -c {$< }

#               *List Macros*

EXE_dependencies =  ¥
  wcsub.obj ¥
  wcount.obj ¥
  a:¥bc¥lib¥wildargs.obj

#               *Explicit Rules*
wcount.exe: wcount.cfg $(EXE_dependencies)
  $(TLINK) /x/c/P-/LA:¥BC¥LIB @&&|
c0s.obj+
wcsub.obj+
wcount.obj+
a:¥bc¥lib¥wildargs.obj
wcount
                # no map file
emu.lib+
maths.lib+
cs.lib+
graphics.lib
|

#               *Individual File Dependencies*
wcsub.obj: wcsub.c

wcount.obj: wcount.c

#               *Compiler Configuration File*
wcount.cfg: wcount.mak
  copy &&|
-v
-vi
-IA:¥BC¥INCLUDE
-LA:¥BC¥LIB
-P-.C
| wcount.cfg
```

リスト4　WCOUNT.C

```
/* =================================================================

    WCOUNT.C            Copyright 1991 by JUSTECH-LABO. (H.kanoshima)

================================================================= */

#include    <stdio.h>

#define     LINEMAX     1024

void    wcount_file(char *fname)
{
FILE    *fp;
static  char    readline[LINEMAX];
char    *rptr , *sptr;

    fp = fopen(fname , "r");
    if(fp == NULL)
    {
        printf("'%s' がありません¥n",fname);
        return;
    }

    printf("=== %s ===¥n",fname);

    while((rptr = fgets(readline , LINEMAX , fp)) != NULL)
    {
    extern  char    *wget(char **);
    extern  void    wset(char *);

        while((sptr = wget(&rptr)) != NULL)
            wset(sptr);
    }
}

void        main(int argc,char **argv)
{
extern  void    dump_word(void);

    while(--argc)
        wcount_file(*++argv);
            dump_word();
}
```

リスト5　WCSUB.C

```
/* =================================================================

    WSUB.C          Copyright 1991 by JUSTECH-LABO. (H.kanoshima)

================================================================= */
#include    <stdio.h>
#include    <stdlib.h>
#include    <string.h>

#define     WORDMAX     1000

typedef     enum    {BFALSE,BTRUE}  BOOL;

struct  _wlist{
    int     wcount;
    char    *wbody;
}wlist[WORDMAX];

struct  _wlist  *wlptr = wlist;

/* ===============================================================


=============================================================== */
static  BOOL    wexist(char *sptr)
{
struct  _wlist  *wlp;

    for(wlp = wlist ; wlp < wlptr ; wlp++)
    {
        if(strcmp(wlp->wbody , sptr) == 0)
        {
            wlp->wcount++;
            return  BTRUE;
        }
    }
    return  BFALSE;
}

/* ===============================================================


=============================================================== */
void    wset(char *sptr)
{
char    *cptr;

    if(wexist(sptr) == BTRUE)
    {
        putchar('!');
        return;
    }
    if(wlptr >= &wlist[WORDMAX])
    {
        printf("単語数が %d 個を超えました。¥n",WORDMAX);
        exit(1);
    }
    cptr = (char *)malloc(strlen(sptr) + 1);
    if(cptr == NULL)
    {
        printf("メモリが足りないので単語リストが作れません¥n");
        exit(1);
    }
    wlptr->wbody = strcpy(cptr , sptr);
    wlptr->wcount = 1;
    wlptr++;
    putchar('.');
}
/* ===============================================================


=============================================================== */
```

ワンラインNEWS
東京特殊電線は同社初のコンピュータ用ディスプレイを開発した．現在のCRTディスプレイと互換があり，従来のパソコンにそのまま接続できる．

区切って複数の指定ができる．全ファイルの単語リストを作成してソートが完了するまでなにごとも起こらないのは寂しいので，動作経過を知らせるために初出の単語が出てきたときは「.」，既出の単語の場合は「！」を表示するようにしている．動作例として，このプログラム自身のソースファイル「wcount.c」を処理させた場合の画面を図8に示す．

なお，今回のバージョンでは，単語リスト構造体の配列数を1000としたため，処理ファイル中に含まれる単語の総数が1000種類を越えた場合は，エラーメッセージを表示して，そこで中断するようになっている．また，1000種類に満たない場合でも，単語の各文字列を格納するメモリ領域が獲得できない場合は中断する．

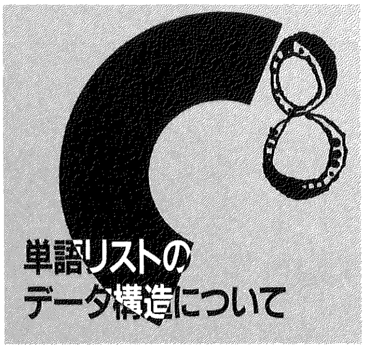
以降，このプログラムの単語リストのデータ構造と各関数のコーディングを1つずつ説明していく．Cのソースになじみの深い読者にとっては少々繁雑になるかもしれないが，初回ということでプログラムの動作をよく理解するためにご勘弁願いたい．

図8　WCOUNT実行画面

```
A>WCOUNT WCOUNT.C
=== wcount.c ===
........................!.........!........!.!.!..!!.!..!!...!.!!.!!!..!.!!!....!
#### ソート中 ####
(0002) - '!='
(0001) - '"r");'
(0001) - '#define'
(0001) - '#include'
(0001) - '%s'
(0001) - '(H.kanoshima)'
(0001) - '*);'
(0001) - '**);'
(0001) - '**argv)'
(0001) - '*/'
(0001) - '*fname)'
(0001) - '*fp;'
(0001) - '*rptr'
(0001) - '*sptr;'
(0001) - '*wget(char'
(0001) - '/*'
(0001) - '1024'
(0001) - '1991'
(0001) - '<stdio.h>'
(0003) - '='
(0001) - '=='
(0002) - '================================================================'
(0001) - '===¥n"'
(0001) - 'Copyright'
(0001) - 'FILE'
(0001) - 'JUSTECH-LABO.'
(0002) - 'LINEMAX'
(0003) - 'NULL)'
(0001) - 'WCOUNT.C'
(0001) - 'argc'
(0001) - 'by'
(0004) - 'char'
(0001) - 'dump_word();'
(0001) - 'dump_word(void);'
(0003) - 'extern'
(0001) - 'fgets(readline'
(0002) - 'fname);'
(0001) - 'fopen(fname'
(0001) - 'fp'
(0001) - 'fp))'
(0001) - 'if(fp'
(0001) - 'main(int'
(0001) - 'printf("'%s'"'
(0001) - 'printf("==='
(0001) - 'readline[LINEMAX];'
(0001) - 'return;'
(0001) - 'static'
(0004) - 'void'
(0001) - 'wcount_file(*++argv);'
(0001) - 'wcount_file(char'
(0001) - 'wget(&rptr))'
(0001) - 'while((rptr'
(0001) - 'while((sptr'
(0001) - 'while(--argc)'
(0001) - 'wset(char'
(0001) - 'wset(sptr);'
(0004) - '{'
(0004) - '}'
(0001) - 'がありません¥n"'
```

```
void    swap_data(struct _wlist *x , struct _wlist *y , int dwidth)
{
char    *cptrx = (char *)x;
char    *cptry = (char *)y;

    while(dwidth--)
    {
    char    tmp;

        tmp = *cptrx;
        *cptrx++ = *cptry;
        *cptry++ = tmp;
    }
}
/* ===============================================================


=============================================================== */
void    sort_data(void)
{
struct  _wlist  *tmp , *ptr_a ;
int     count , major_count ;
BOOL        swap_flag ;

    if(wlptr == wlist) return;
    major_count = wlptr - wlist - 1;
    while(major_count)
    {
        count = major_count;
        ptr_a = wlist ;
        swap_flag = BFALSE ;
        while(count)
        {
            if(strcmp( ptr_a->wbody , (ptr_a + 1)->wbody) > 0) {
                swap_data(ptr_a, (ptr_a + 1), sizeof(struct _wlist));
                swap_flag = BTRUE ;
            }
            ptr_a++;
            count--;
        }
        major_count--;
        if(swap_flag == BFALSE) return ;
    }
}
/* ===============================================================


=============================================================== */
void    dump_word(void)
{
struct  _wlist  *wlp;
    printf("¥n#### ソート中 ####¥n");
    sort_data();
    for(wlp = wlist ; wlp < wlptr ; wlp++)
        printf("(%04d) - '%s'¥n" , wlp->wcount , wlp->wbody);
}
/* ===============================================================


=============================================================== */
char    *wget(char **sptrptr)
{
static  char    getw[2048];
char    *sptrs , *sptrd;
int     cplen;

    sptrs = *sptrptr;

    sptrs += strspn(sptrs , " ,¥t¥n");
    if(*sptrs == '¥0')  return  NULL;

    sptrd = strpbrk(sptrs , " ,¥t¥n");
    if(sptrd == NULL)   cplen = strlen(sptrs);
    else                cplen = sptrd - sptrs;
    sptrd = getw;
    while(cplen--)
        *sptrd++ = *sptrs++;
    *sptrd = '¥0';
    *sptrptr = sptrs;

    return  getw;
}
```


ワンラインNEWS
リコーはノート型パソコン「RICHO PS55ノート」シリーズに80Mバイト HDD搭載モデルなど4モデルを追加，発表した．従来同様，日本IBMのOEM供給を受けての販売．

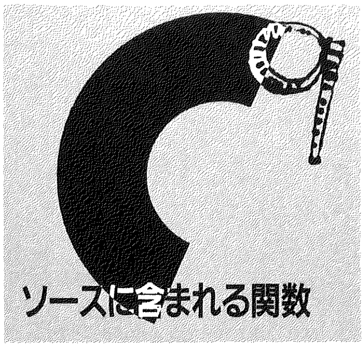


単語リストのデータ構造について

出現した単語の個数を調べるために，単語をリスト化して保存する．このリストのデータ構造は図9のようになっている．

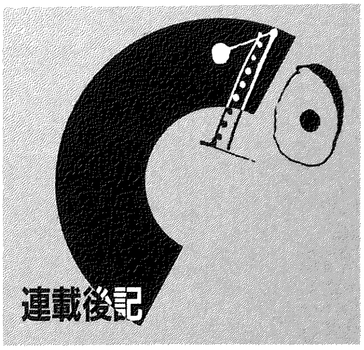
単語リストは，カウンタと文字列へのポインタからなる構造体がWORDMAX個分の配列となった領域である．wlistは，その配列の実体のシンボルである．wlptrは，次に格納するポインタを格納する変数であり，新しく出てきた単語を1つ格納するたびに更新される．wlptrがWORDMAX個目の要素を越えた場合は，これ以上の新出単語は処理できない．

ある単語を処理する場合，まずリストの先頭位置から格納されている文字列と処理対象の文字列を比較していく．一致したものがあれば，既出の単語であるのでカウンタを+1する．wlptrの示す位置まで一致した文字列がなければ，新規登録として処理単語の文字列長だけメモリをアロケートして単語を格納し，その文字列の格納アドレスとカウンタの初期値である1をwlptrの位置に書き込んでwlptrを更新する．



ソースに含まれる関数

表1がメインモジュールwcount.cに含まれる関数である．この部分は，テーマツールの基本仕様として，次回以降も変更はないはずである．実際の処理関数である「wget」「wset」「dump word」の各関数を，新しく出てきたアルゴリズムによって毎回新しいものに置き換えていくことになる．というわけで，今月の「wget」「wset」「dump word」を紹介しよう（表2）．



連載後記

今回は，「ソート」の前編として「選択ソート」と「バブルソート」を紹介した．前のほうでも書いたが，ソートには今回紹介したほかにもさまざまな手法がある．しかも，そのうちのどれがいちばんよいかというのは，データ

図9

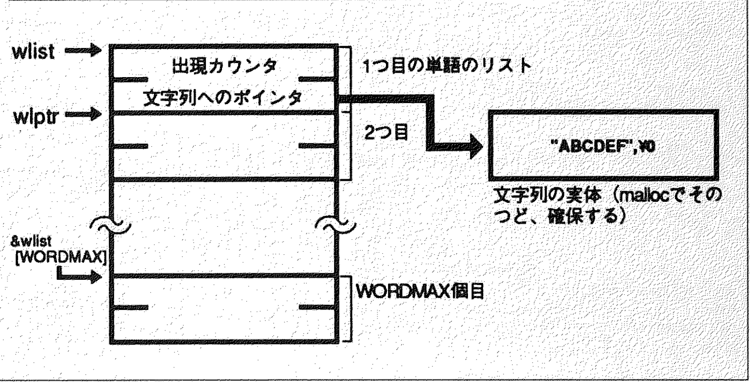


表1　メインソース「WCOUNT.C」に含まれる関数

```
void main(int argc,char **argv)

  : while(--argc) {
  :     wcount_file(*++argv);
  :     dump_word();
  : }
```

起動パラメータで与えられたファイル名を1つずつ切り出して，wcount_file関数に引き渡し，単語リストを作成していく．起動パラメータがなくなったら，結果をdump_word関数によって表示させて処理を終了する．

```
void wcount_file(char *fname)

  : fp = fopen(fname , "r");
  : if(fp == NULL)
  : {
  :         printf("'%s' がありません¥n",fname);
  :         return;
  : }
```

渡されたファイル名でファイルオープンを実行する．オープンできなかったら，エラーメッセージを表示して処理を終える．

```
  :         printf("=== %s ===¥n",fname);
```

オープンしたファイル名を表示する．

```
  :         while((rptr = fgets(readline , LINEMAX , fp)) != NULL)
```

ファイルから1行読み込みを実行し，データがなくなるまで以下を繰り返す．

```
  : while((sptr = wget(&rptr)) != NULL)
  :     wset(sptr);
```

wget関数を実行して単語を切り出し，wset関数によって単語リストに登録していく．この処理を1行分繰り返す．

ワンラインNEWS
ソニーはパソコンやワークステーション用の外部記憶装置である光磁気ディスク装置「RMO-S550」を発売した．従来製品にくらべ機能強化，小型軽量化を進めている．

の性質や量などの要素によって変わってくるため一概には決められない．次回も引き続いてソートアルゴリズムをいくつか紹介しながら，このあたりの話を進めていきたい．そして，より効率のよいソートのロジックを用いて，テーマツールの高速化を試みる予定である．

表2　サブソース「WCSUB.C」に含まれる関数

```
char  *wget(char **sptrptr)
```

入力：元文字列へのポインタ
出力：切り出した単語の格納位置へのポインタ
　　　元文字列へのポインタは切り出した単語の次の文字をさしている．

```
:sptrs = *sptrptr;
```
渡されたポインタを保存する．

```
: sptrs += strspn(sptrs , " ,\t\n");
```
文字列の先頭から半角の空白，「，」，ＴＡＢ，改行のいずれか以外の最初の文字が出てくるまでポインタを進める．実際には行頭の空白をスキップする処理になるだろう．

```
: if(*sptrs == '\0')  return NULL;
```
行頭の空白などを取り除いた結果，文字列が空になった場合はNULLを返して処理を終了する．

```
: sptrd = strpbrk(sptrs , " ,\t\n");
```
半角の空白，「，」，ＴＡＢ，改行のいずれかが最初に出現するポインタを求める．この４種類が単語の区切りとなる．

```
: if(sptrd == NULL)   cplen = strlen(sptrs);
: else  cplen = sptrd - sptrs;
```
単語の区切りが得られなかった場合は，行末までが１つの単語とみなして文字列全体の長さを，得られた場合は「区切り位置 － 文字列先頭」の長さを単語長とする．

```
: sptrd = getw;
```
単語格納位置のポインタをセットする．

```
: while(cplen--)
: *sptrd++ = *sptrs++;
```
単語長だけ元文字列からコピーする．

```
: *sptrd = '\0';
```
単語の末尾にヌルを付加する．

```
: *sptrptr = sptrs;
```
元文字列へのポインタを切り出した単語の直後へ更新する．

```
: return getw;
```
単語の格納エリアのポインタを返す．

```
void  wset(char *sptr)
```

入力：単語へのポインタ
出力：なし

```
:  if(wexist(sptr) == BTRUE) {
:     putch('!');
:     return;
:  }
```
wexist関数を使って，単語が既出のものかどうか判定する．wexist関数は
　・単語が初出であれば，BFALSEを返す．
　・単語が既出のものであれば，カウンタを+1してBTRUEを返す．
という仕様になっている．既出の単語だったときは「！」を表示する．

```
:  if(wlptr >= &wlist[WORDMAX])
:  {
:     printf("単語数が %d 個を超えました．\n",WORDMAX);
:     exit(1);
:  }
```
新規単語を登録するために，単語リストに空きがあるかどうかチェックする．格納ポインタ（wlptr）が構造体の最終要素（&wlist[WORDMAX]）を越えていたら空きがないと判定して，エラーメッセージを表示し，プログラムを強制終了する．

```
:  cptr = (char *)malloc(strlen(sptr) + 1);
:  if(cptr == NULL)
:  {
:  printf("メモリが足りないので単語リストが作れません\n");
:  exit(1);
:  }
```
新規の単語の文字列を格納するための領域をアロケートする．サイズを+1しているのは，文字列終端のヌルの分である．もしアロケートが失敗したら，これ以上のリストを作成できないため，エラーメッセージを表示し，プログラムを強制終了する．

```
:  wlptr->wbody = strcpy(cptr , sptr);
```
アロケートした領域に単語をコピーする．strcpy関数は，コピー先のポインタをそのまま返してくるので，それをリストにセットする．

```
:  wlptr->wcount = 1;
```
単語リストのカウンタ初期値として１をセットする．

```
:  wlptr++;
```
単語リストのポインタを更新する．++は，アドレスを+1するのではなく，構造体の要素数に対する+1なので注意すること．

```
:  putch('.');
```
新規単語を登録したときは「．」を表示する．

```
void dump_word(void)
```

入力：なし
出力：なし

```
: printf("\n**** ソート中 ****\n");
```
ソート開始のメッセージを表示する．

```
: sort_data();
```
単語リストをソートする．詳細は後述．

```
: for(wlp = wlist ; wlp < wlptr ; wlp++)
:       printf("(%04d) - '%s'\n" , wlp->wcount , wlp->wbody);
```
単語リストの先頭から，実際に格納した位置までの数だけ「－」をはさんで単語のカウンタと文字列を表示する．

```
void sort_data(void)
```

入力：なし
出力：なし

```
: if(wlptr == wlist) return;
: major_count = wlptr - wlist - 1;
```
ソートするデータの総数は，単語リスト構造体の配列要素数である．

```
: while(major_count)
```
外側のループ

```
: {
:       count = major_count;
```
内側のループ回数の初期値

```
:       ptr_a = wlist ;
```
常に先頭の構造体から処理する

```
:       while(count)
```
内側のループは１要素ずつ処理回数が減っていく

```
:       {
:            if(strcmp( ptr_a->wbody , (ptr_a + 1)->wbody) > 0)
```
文字列同士の比較を行う

```
:                 swap_data(ptr_a, (ptr_a + 1), sizeof(struct _wlist));
```
入れ替えが必要な場合は，swap_data関数を使って入れ替える．

```
:            ptr_a++;
```
比較対象を１つ後ろへ

```
:            count--;
```
内側のループカウンタの更新

```
:       }
:       major_count--;
```
外側のループカウンタの更新

```
void swap_data(struct _wlist *x , struct _wlist *y , int dwidth)
```

入力：交換するデータへのポインタ１，２，交換するデータ長
出力：なし

```
: while(dwidth--)
: {
:       tmp = *cptrx;
:       *cptrx++ = *cptry;
:       *cptry++ = tmp;
: }
```
データ長分だけ，ローカル領域を使用しながらデータ１と２を入れ替える．

ワンラインNEWS
ヤマハは米ラステック社と共同で，ページプリンタの制御回路向けLSI「GC1002」を開発した．このチップを使うと小型にできるほか高速な処理を可能とする．

98活用アセンブラ技法

SCSIドライバ編 第1回

SCSIデバイスドライバ/スケルトンの作成

星野 操

この98活用アセンブラ技法の連載では，おもにメモリ常駐終了型（TSR）のプログラムを作成してきたが，今回より数回に渡ってデバイスドライバ型のプログラムを作成する．

デバイスドライバを作成する場合にも，常駐プログラム同様スケルトンがあると，プログラミングが容易になる．そこで今回は，デバイスドライバのスケルトンとSCSIデバイスドライバを作成する．

MS-DOS 5.0のデバイスドライバ

デバイスドライバに先だってTSRを常駐させることも可能

前回までのBIOSのすり替えなどによる機能拡張においては，主としてTSRを使用してきました. 11月末には98用のDOS5もリリースされると伝えられています. DOS5(DOS4から)だと, CONFIG. SYSの中でINSTALL＝という行が使用でき，これを用いるとデバイスドライバに先立って常駐プログラムを常駐させることが可能となります.

これによって，以前にはデバイスドライバを使用しなければいけなかった場合にも，常駐プログラムを使用することが可能となります．ますます常駐プログラムの有効性が高まるというわけですが，もちろんデバイスドライバのすべてを常駐プログラムに置き換えるというのはあまり現実的ではありません.

やはり，本当にデバイスとして使用すべきものは，デバイスドライバとして書くほうが自然なのは確かです．今回から少し，デバイスドライバ形式でのプログラムについてふれてみることにしましょう. まず，今回はSCSIデバイスのためのドライバというものを書いてみることにします.

常駐プログラムは，高級言語でもかなりの部分まで記述することができます. interrupt修飾子が導入されてから，常駐サイズさえ気にしなくてよいならアセンブラはいっさい使用せずに高級言語だけで記述することも可能です．これに対して，デバイスドライバはそのように簡単にはいきません.

ロードされるドライバの先頭部分には，特定のシステムのためのデータ領域が必要です．これを高級言語で書くことは，どうしても不可能です．少なくとも，アセンブラで書かれているスタートアップルーチンを書き直さなければいけません．つまり，デバイスドライバを書く上では，どうしてもアセンブラとかかわりをもたないわけにはいかないのです.

キャラクタデバイスとブロックデバイス

デバイスの差による2つの形式

デバイスドライバには，ブロック形式とキャラクタ形式の2つの形式があります．前者は512バイトなどのブロックを単位として読み書きするもので，後者は1バイト単位で読み書きをするものです．これらには，いくつかの大きな違いがあります.

まず，キャラクタデバイスには名前がありますが，ブロックデバイスにはありません. このことと関連しますが，キャラクタデバイスは同名のものを複数リンクすると，あとからリンクされたものが有効になりますが，ブロックデバイスは単に新しいドライブがあとに追加されるだけです．このため，キャラクタデバイスはADDDRVなどでデバイスリンクに接続すれば比較的簡単に後ろからドライバをくっつけることも可能ですが，ブロックデバイスの場合はそのようにはいきません.

次に，ブロックデバイスはダイレクトアクセスデバイスですが，キャラクタデバイスはシーケンシャルデバイスです．このため，ブロックデバイスはどこを読みにいくのかというブロックの「通し番号」を教えてやる必要がありますが，キャラクタデバイスにはその必要はありません．そのかわりキャラクタデバイスの場合は，ブロックデバイスには必要ない入出力ビジー状態の監視や，入出力バッファのフラッシュ管理などが必要になってきます．キー入力がきているか，プリンタ出力が終了しているかを監視したり，入力を読み捨てなければいけない場合などが存在するからです．ブロックデバイスでは，ダイレクトアクセスですから，そのような問題は発生しません.

ブロックデバイスの容量制限

ここで，ブロックデバイスの容量制限とクラスタサイズの制限についても触れておくことにしましょう．前記のように，ブロックデバイスはダイレクトアクセスデバイスですから，アクセスする位置を知らせてやらないといけません．つまり，通しのセクタ番号を教えてやる必要があるのです. ここで，セクタ番号はDOS3. Xまでは16ビットで示されていました. DOSはこれによって最大65536セクタにしかアクセスできないことになります. もちろん，この中にはデータとして使われる領域のほかに，システムが使用するデータ領域やFAT／ディレクトリの領域も含めなければいけません.

DOSでは, デフォルトのバッファを起動時に取ることから，デフォルトのブロックデバイスのセクタサイズ以上のブロックデバイスを使用することはできません. IBM系のシステムでは，すべてのブロックデバイスは512B／セクタに統一されています. 98では，8インチドライブの関係から，1KB／セクタが一般的に使用されます.

結果としてIBM系のドライブは32MB，98では64MBが(論理)ドライブ容量の最大値となります. 最近のDOSでは, 128MBまでのパーティションもサポートされていますが，64MB以上のパーティションを作ったときにはセクタサイズが2KBになります. これに

ワンラインNEWS
クレオはPC/AT上で稼働する日本語ワープロソフト「漢字コム」を開発，北米で販売を開始した．第2弾として同社の人気ワープロソフト「ユーカラ」も販売予定.

よって128MBを65536セクタの制限を保ったままアクセス可能とするのです．

内部にパッチを当てたりすれば，さらに大きなセクタサイズを構成することも可能となります．たとえば8KB／セクタとすれば，512Mまで1つのドライブで取ることができます．MOのディスク片面をパーティションを切らずに使用できるわけです．

この代償として，システムで確保するBUFFERSのサイズが大きくなります．たとえば標準的な20個のディスクバッファを確保した場合には，2KB／セクタであっても実に40KBものメモリを消費してしまうことになります．これは馬鹿になりません．

この問題を解決するのが，DOS4から導入された22ビットセクタです．つまりセクタサイズを表現するのに，32ビットの領域を使用するのです．もちろん既存のドライバも使えないと意味がないので，いろいろコンパチビリティを保つための工夫をしています．本来なら，32ビットまるまるセクタ番号に使えてもよいのですが，実際には22ビット（DOS5では23ビットという話）までしか有効ではないようです．これはなんらかの理由により，ドライブの最大容量とファイルの最大容量とを同一にするためでしょう．

DOSでは，ファイルサイズは32ビットで表現されます．つまり，2GBまで1つのファイルがもてるわけです．一方，ディスク容量のほうは，22ビットのセクタ番号と9ビット（512バイト）のセクタサイズとで符合付き31ビットバイトの容量になるというわけです．98の場合は1KBなので，2GB／ドライブとなります．さらにセクタサイズを増やしたとしても，2GBの壁は超えられないのではないかと思います．

ディスクの直接アクセスでは，INT25などの割り込みが使用されています．これらもセクタ番号は16ビットのレジスタ1個で渡すようになっています．これではDOS4以降の拡張されたブロックデバイスに対応できないので，この場合には拡張されたセクタ番号を格納するメモリのアドレスを渡すようになっています．

FATの制限

以上のようにドライブの容量については改善がなされました（もっとも，最近では2GB程度の容量は容易に確保できるようになってしまいましたが），もう1つFATの制限というものを考えなければいけません．

DOSのディスク管理だと，ディスクの中身はクラスタという単位に分割されて管理されます．このクラスタの管理状態を示すテーブルがFATです．以前のDOSはFATのエントリを12ビットで表現していました．つまり1つのドライブは4096ぐらいのクラスタしかもつことができませんでした．

仮に40MBのドライブの場合，1KB／セクタで40000余りのセクタをもっています．これを4096のクラスタに収めるわけですから，1つのクラスタで10以上のセクタをまとめなければいけません．実際には，クラスタは2^N数のセクタしか扱うことができないので，1クラスタ＝16KB単位となってしまいます．つまり，どんな小さなファイルを作っても必ず16KBが消費されてしまうということです．

これでは不経済なので，最近ではほとんどの場合16ビットでクラスタエントリを表現する方法が取られています．つまり65536程度のクラスタをもつことができるわけですから，40MBでも1クラスタ＝1セクタ＝1KBで表現することが可能となります．DOS3.Xではセクタ数とクラスタ数の表現能力が同一ですから，どのようなドライブであっても1クラスタ＝1セクタとすることが可能です．

ただし，実際にはもう1つ制限があります．1エントリを表現するのに，16ビット-FATだと2バイト必要になります．ですから，FATのエントリを全部使用すると，128KBのFATサイズになります．普通にファイルを扱っている場合には問題になりませんが，FAT自体を処理する場合，たとえばCHKDSKのようなプログラムだとこのためのバッファを64KBしか用意していない場合があります．バッファが複数のセグメントにまたがると，扱いにくいということでしょう．

このような制限があると，実際に使用できるクラスタ数は，物理的制限の半分になってしまいます．前記の40MBのドライブの場合には，1クラスタ＝2セクタとなってしまうわけです．

さてDOS4.X以降になると，やはりセクタ数のほうがクラスタ数よりも大きくなります．このため1つのクラスタに複数のセクタをまとめる必要がでてきます．仮に，512MBのパーティションを切ったとしましょう．これを1KB／セクタで表現すると，512K個のセクタが必要になります．クラスタ数の制限は64K個または32K個ですから，1クラスタに8または16セクタが必要になります．結局1セクタ＝8KBまたは16KBとなるわけです．2KBセクタで表現したとしても，やはり1クラスタの大きさは変わりません．ですから，22ビットセクタのドライバだと，セクタサイズの決定はFATの拡大効果によるスピードダウンのデメリット

と，バッファサイズ縮小のメリットのトレードオフということになります．つまり，22ビットセクタになったとしても，ディスクが無条件に有効に使用できるというわけではないのです．

よくDOS4では32ビット-FATが導入されたと誤解されている人がいますが，DOS4で拡張されたのはセクタ番号であるということに注意しなければいけません．

話を元に戻しますが，これらのアクセス方法の違いにより，キャラクタデバイスとブロックデバイスとでは用意されているファンクションが違います．目的を達するために，たとえばキャラクタデバイスには，読み込みバッファをそのままにして入力データを取ってくるというファンクションなどがあります．

デバイスドライバを作成する場合にも，常駐プログラムと同様にスケルトンがあるとずっと簡単になります．スケルトンには，このブロックデバイスとキャラクタデバイスとで異なる部分は含めないようにし，意外と書くのがめんどうなオプションの解析ルーチンなどを入れておくことにします．

今回のプログラム
スケルトン＆SCSIドライバ

今回のプログラムは，3重構造になっています．まずいちばん外側にデバイスドライバのスケルトンであるDEVSTD.ASMがあります．これには前記のように枠組といくつかのユーティリティが入っています．このファイルは，さらにXSCSI.ASMをインクルードします．デバイスドライバに必要な実際の処理機能は，このインクルードされるファイルに記述することになるわけです．さらに，XSCSIはNECSCSI.INCをインクルードします．

これらによって作成されたXSCSI.SYSの目的は，統一された手続きでのSCSIインターフェースを提供することです．98のSCSIボードでは，比較的使いやすいSCSI-BIOSが提供されます．しかし，これにしても何も考えずに使えるというわけではありません．たとえばエラーが発生したときの処理など，BIOSの機能を複合して使用しなければいけない場合があります．

これが他機種の場合やオリジナルなボード(SPCボードなど)を使用する場合，SASIボードを流用する場合など，適切なサービスが提供されていない場合もあります．SCSIのいちばん基本的なコマンド送出とメッセージの受け渡しの部分はハードウェア依存になってしまいます．この部分をまず切り分け，ハードウェアの差異を吸収します．98のSCSIボードを使った場合というのが，このNECSCSI.INCにあたります．

さらにその上にSCSIのアクセスをする，機種非依存なインターフェースを設けます．これがXSCSI.ASMです．そしてそれをデバイスドライバとして仕立てあげるために，スケルトンDEVSTD.ASMを用いるのです．

デバイスドライバをほかの目的で使用する場合，たとえば次回のようにテープドライバなどを構成する場合には，2番目のレベル，すなわちXSCSI.ASMに相当するファイルを別に用意することになります．

DEVSTD.ASM

では，まずDEVSTD.ASMから見ていきましょう．

デバイスドライバでは，基本的にcomモデルのほうが便利です．exeモデルでも記述することは可能ですが，サイズが小さい場合にはそれほどのメリットはありません．このため，今回は以前の例とは違って簡易セグメント指定は用いていません．

また，オープンな手続きが多いので，これらはPROCという形を用いずラベルを使用しています．PROCにしても，オーバーヘッドなどを嫌ってfuncマクロは使用していません．

プログラムの先頭では，CONDX.INCを読み込んだ後，通常のセグメント指定を行っています．CS／DSともにセグメントCODEに割り当てることにします．次のbios_idというのは，このSCSIインターフェースプログラムと，それを利用する別のプログラムが通信するためのキーで，XSCSI.SYSでサポートされるデバイス名を意味します．

デバイスドライバの先頭部分には，デバイスヘッダというものが必要になります．これについての詳細は，別に参考書を参照してください．ここでは，3つのモードを用意しています．PRIME_DEVが定義されていると，XSCSIを作成するために適した設定になります．これはダミーのキャラクタデバイスです．TAPE_DEVが設定されていると，テープのキャラクタデバイスとなります．さらに，BLOCK_DEVが設定されていると，ブロックデバイスの設定になります．

デバイスヘッダは，これら3つのためのものが用意されています．TAPE_DEVの場合には，OPEN／CLOSEなどの拡張機能がサポートされているということを示すフラグが設定されています．ブロックデバイスの場合は，キャラクタデバイスの名前にあたる部分にサポートしているドライブ数を入

ワンラインNEWS
エイサーの日本法人日本サーテックは米シャイアン・ソフトウェアと提携し，LAN用バックアップソフト「アークサーブ」の日本語版を開発する．

れるようにという指示があり，このプログラムでもそれに従っていますが，実際にシステムで使用されることはないようです.

次に，デバイスドライバストラテジールーチンで渡されるコマンドパケットの位置を記憶しておくポインタと，パケットの構造を示す構造体が定義されています．その後ろにあるワークはこのプログラムでは使用しませんが，XSCSIを呼び出すときのエントリポイントを格納するのに使用します.

次にあるのが，ドライバが返すエラーコードの一覧定義です．その後ろにストラテジールーチンがあり，ここでは単にパケットのアドレスをポインタに格納しています．COMモデルなので，DSの設定は省き，CSオーバーライドで簡単に対処しています.

その次には，コマンドパケットによって渡されたコマンドコードに対応するエントリのテーブルがあります．ここで定義されているエントリは，すべてインクルードされるファイルの中で定義されていないといけません.

そしてインタラプトルーチンがあります．ここでは，全レジスタをセーブした後，DSの値をセットアップします．直接CODEセグメントの値を参照するとCOM化できなくなりますから，CSの値を用いてDSを設定しているわけです．それからコマンドコードが正常な範囲であるかどうかを調べ，正常であればテーブルによってジャンプします．異常であれば，error_commandに飛びます．これもやはり，インクルードされるファイルで定義されないといけません.

そしておのおのの処理が終わったあとはinterrupt_endに処理が移るようにし，ステータスを記憶した後レジスタを復元して復帰します．ここで，ローカルスタックを用意してそれを設定するということは行っていません．デバイスドライバなどでは全レジスタの保存ぐらいは問題ないが，それ以上スタックを使う場合にはローカルスタックを用意しろといわれています．しかし，現実にどれだけのローカルスタックを取ればよいかというのは大きな問題です．いろいろなドライバや常駐プログラムと併用することを考えると，実際問題として256バイトでは小さ過ぎることが少なくありません．512バイトでもトントンぐらい，できれば1024バイトぐらい欲しくなります.

デバイスドライバが呼ばれるときには，ユーザープログラムのスタック領域は使用されず，DOSの内部ワークエリアに取られたスタックが使用されます．これの大きさがどの程度であるかというのはよくわからないのですが，あまり小さいものではないようです．経験的に，デバイスドライバで使用する程度なら，多少大目にスタックを使ったとしてもローカルスタックを用意するよりそのまま渡されたスタックを使ったほうがよい結果が得られます.

ここまで述べてきた部分はデバイスドライバが常駐している最中に必要なルーチンです．それ以外のオプション解析や初期化などのルーチンは，初期化が終了すると不要になります．これらの部分は常駐プログラムの場合と同様，切り放すことができます．そこで，このすぐ後ろにインクルードファイルを読み込み，その中で定義される（今回の場合，NECSCSIの中）end_labelというラベルの前までを常駐域に入れることにします.

その次のsearch_biosというのは，このXSCSIを利用するドライバが，XSCSIの存在を確認し，そのエントリルーチンをワークエリアに格納するためのものです.

最初に，XSCSIのダミーのキャラクタデバイスをオープンします．このデバイス自体は読み書きに使用されることはありませんが，そのかわりにIOCTLをサポートしています．デバイスドライバに対しては，IOCTLコマンドを利用することによって，インターフェースを取ることができます．常駐プログラムでINT2Fが標準のインターフェース手段であるのと同様，デバイスをオープンしてからIOCTLを実行するというのがデバイスドライバの標準的な通信手段です.

正常にオープンできたら，IOCTLを利用し，4キャラクタのデータを読み込みます．返される値は，XSCSIのエントリテーブルのアドレスです．これから，各エントリアドレスを計算し，ワークエリアに設定をします．この関数並びにワークエリアは，今回は使用されません.

その次にあるのが，オプションパーザーとオプションチェッカです．ここでは，解析されたオプションは構造体_optionの配列によって示されます．最初のデータがオプションキャラクタで，次が引数へのlongポインタです.

atohはオプションを利用するときに使用するユーティリティで，引数のhex文字をデータに変換するときに使用します．アドレスはES：BXから取られます．値はAXに返されDLが破壊されます．parse_optionはオプションパーザーです．まずコマンドパケットからオプション文字列へのポインタを取り出し，それの終端までスキャンします．／が現われたら，その次の文字をオプション文字と解釈し，その文字の次のアドレスとともにワークエリアに格納します．オプションが多過ぎる場合には，処理を中断します.

ワンラインNEWS
日立製作所は記憶容量が世界最大の3.5インチHDD「DK315Cシリーズ」と5インチHDD「DK517Cシリーズ」の販売を開始した．記憶容量は各1.4/3.7Gバイト.

リスト1 DEVSTD.ASM

```
;-----------------------------
; device driver common routine
;-----------------------------
        masm51

        include condx.inc

code    segment public
        assume  cs:code, ds:code

bios_id equ     <'X$SCSI$X'>

; device header
device_header:
ifdef   PRIME_DEV
        dd      -1                      ; link pointer
        dw      0c000h                  ; character device, ioctl
        dw      strategy
        dw      interrupt
        db      bios_id
elseifdef       TAPE_DEV
        dd      -1                      ; link pointer
        dw      0c800h                  ; character device, ioctl
        dw      strategy
        dw      interrupt
drive_number    label   byte
        db      'STAPE   '
else
        dd      -1
        dw      0
        dw      strategy
        dw      interrupt
drive_number    db      ?
endif

; packet work
packet  dd      ?

packet_header   struc
len     db      ?
unit    db      ?
cmd     db      ?
status  dw      ?
        db      8 dup (?)
media   db      ?
trans   dd      ?
count   dw      ?
start   dw      ?
regstered_drive         db      ?
packet_header   ends

; bios entry pointer
ifndef  PRIME_DEV
read_write_entry        dd      ?
get_capacity_entry      dd      ?
scsi_prime_entry        dd      ?
sense_buffer_address    dd      ?
endif

; error code
e_write_protect equ     8100h
e_unknown_drive equ     8101h
e_not_ready     equ     8102h
e_bad_command   equ     8103h
e_crc_error     equ     8104h
e_bad_length    equ     8105h
e_seek_error    equ     8106h
e_invalid_media equ     8107h
e_no_sector     equ     8108h
e_paper_out     equ     8109h
e_write_error   equ     810ah
e_read_error    equ     810bh
e_error         equ     810ch
e_bad_change    equ     810fh

; strategy routine
strategy:
        mov     word ptr cs:packet,bx
        mov     word ptr cs:packet+2,es
        retf

; command table
command_table   label   word
        dw      init
        dw      media_check
        dw      build_bpb
        dw      ioctl_in
        dw      read
        dw      nd_read
        dw      read_status
        dw      read_flush
        dw      write
        dw      write_verify
        dw      write_status
        dw      write_flush
        dw      ioctl_out
        dw      device_open
        dw      device_close
        dw      removable_media
        dw      output_until_busy
        dw      deinstall
command_table_tail      label   word

; interrupt routine
; save registers
interrupt:
        push    ax
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        push    ds
        push    es
; get command code
        push    cs
        pop     ds

        les     bx,packet
        mov     al,es:[bx].cmd
        cmp     al,(command_table_tail-command_table)/2
        jnc     error_command
; jump to table
        mov     ah,0
        add     ax,ax
        mov     bx,ax
        jmp     command_table[bx]

interrupt_end:
        les     bx,packet
        mov     es:[bx].status,ax
        pop     es
        pop     ds
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        retf

ifdef           BLOCK_DEV
        include xdrive.asm
elseifdef       TAPE_DEV
        include xtape.asm
else    ;       PRIME_DEV
        include xscsi.asm
endif

;----------------------------
; from here, non resident part
;----------------------------

; bios entry searcher
ifndef  PRIME_DEV
bios_id_string  db      bios_id,0
bios_table_buf  dd      ?

; search bios entry, es:bx for entry table
search_bios     proc
        ; open device
        mov     ah,3dh
        mov     dx,offset bios_id_string
        mov     al,0
        int     21h

        ; check status
        @if     c
                ret                     ; error return
        @endif

        ; get ioctl
        mov     bx,ax
        mov     ax,4402h
        mov     dx,offset bios_table_buf
        mov     cx,4
        int     21h

        ; check status
        @if     c
                ret                     ; error return
        @endif

        ; close handle
        mov     ah,3eh
        int     21h

        ; set work area
        les     bx,bios_table_buf
        mov     ax,es:[bx]
        mov     word ptr read_write_entry,ax
        mov     word ptr read_write_entry+2,es
        mov     ax,es:[bx+2]
        mov     word ptr get_capacity_entry,ax
        mov     word ptr get_capacity_entry+2,es
        mov     ax,es:[bx+4]
        mov     word ptr scsi_prime_entry,ax
        mov     word ptr scsi_prime_entry+2,es
        mov     ax,es:[bx+6]
        mov     word ptr sense_buffer_address,ax
        mov     word ptr sense_buffer_address+2,es
        clc
        ret
search_bios     endp
endif

; for option parser
_option struc
o_char  db      ?
o_arg   dd      ?
_option ends

options _option 20 dup (<0,0>)
options_tail    label   byte
        _option <0,0>

```


ワンラインNEWS
米サンタ・クルーズ・オペレーションはOAGDの規格に対応したパソコン用UNIXの開発を開始した。サン・マイクロシステムズ社も同様のソフトを日本市場に投入する。

ここで解釈されたオプションは，ドライバの初期化部分ですべて理解され使用されなければいけません．使用されていないオプションが与えられていることを警告するのが，check_optionです．使用されたオプションは，そのオプション文字の位置を0でクリアします．クリアされていない要素を発見した場合には，その旨メッセージを出して警告します．

そして最後の部分が，スケルトンの初期化部分です．最初に，インクルードされるファイルの中で定義される，scsi_mesという文字列を出力します．それから，XSCSIでない場合には，XSCSIのデバイスを探し，見つからないとその旨警告してインストールを中断します．

インストールを中断するためには，デバイスをブロックデバイスとして扱い，サポートするドライブ数を0とします．

そのオプションの解析を行い，次にインクルードされるファイルの初期化ルーチンを呼びます．そして処理されていないオプションのチェックをしてから常駐域を示すアドレスの設定をし

```
; ascii to hex, pointer is es:bx
atoh    proc
        sub     ax,ax
        @begin
                mov     dl,es:[bx]
                inc     bx
                @mc
                        cmp     dl,'0'
                        @cond   ae
                        cmp     dl,'9'
                        @cond   be
                @if
                        ; 0-9 number
                        sub     dl,'0'
                @else
                        or      dl,20h
                        @mc
                                cmp     dl,'a'
                                @cond   ae
                                cmp     dl,'f'
                                @cond   be
                        @if
                                ; a-f number
                                sub     dl,'a'-10
                        @else
                                @break
                        @endif
                @endif

                add     ax,ax
                add     ax,ax
                add     ax,ax
                add     ax,ax
                or      al,dl
        @again
        ret
atoh    endp

too_many_mes    db      'Too many options',0dh,0ah,'$'

parse_option    proc
        ; get argument pointer
        les     bx,packet
        les     bx,dword ptr es:[bx].count

        ; parse it
        mov     si,offset options       ; pointer
        @begin
                ; get argument character
                mov     al,es:[bx]
                inc     bx

                ; search tail
                cmp     al,0dh
                @if     z
                        @break
                @endif
                cmp     al,0ah
                @if     z
                        @break
                @endif

                ; check option key
                cmp     al,'/'
                @if     nz
                        @break
                @endif

                ; option code
                mov     al,es:[bx]
                or      al,20h
                mov     [si],al
                inc     bx
                mov     [si+1],bx
                mov     [si+3],es
                dec     bx
                add     si,5
                cmp     si,offset options_tail
                @if     ae
                        ; too many options
                        mov     ah,9
                        mov     dx,offset too_many_mes
                        int     21h
                        ret
                @endif
        @again
        ret
parse_option    endp

not_parse_mes   db      'Not parsed option:$'

; check if all options parsed
check_option    proc
        mov     si,offset options
        @begin
                cmp     si,offset options_tail
                @if     ae
                        @break
                @endif

                cmp     [si].o_char,0
                @if     nz
                        ; not parsed option
                        mov     ah,9
                        mov     dx,offset not_parse_mes
                        int     21h
                        mov     ah,2
                        mov     dl,[si].o_char
                        int     21h
                        mov     dl,0dh
                        int     21h
                        mov     dl,0ah
                        int     21h
                @endif
                add     si,5
        @again
        ret
check_option    endp

not_bios_mes    db      'No bios device installed',0dh,0ah,'$'

init:
        ; message output
        mov     dx,offset scsi_mes
        mov     ah,9
        int     21h

ifndef  PRIME_DEV
        ; search device
        call    search_bios
        @if     c
                ; message output
                mov     dx,offset not_bios_mes
                mov     ah,9
                int     21h

                ; never install
                mov     word ptr ds:[4],0
                les     bx,packet
                mov     word ptr es:[bx].trans,0
                mov     word ptr es:[bx].trans+2,cs
                mov     drive_number,0
                mov     es:[bx].media,0
                mov     ax,100h
                jmp     interrupt_end
        @endif
endif

        ; option parser invoke
        call    parse_option

        call    ex_init

        call    check_option

        les     bx,packet
        mov     word ptr es:[bx].trans,offset end_label
        mov     word ptr es:[bx].trans+2,cs
ifdef   BLOCK_DEV
        mov     al,drive_nubmer
        mov     es:[bx].media,al
        mov     es:[bx].count,offset bpb_array
        mov     es:[bx].start,cs
endif
        mov     ax,100h
        jmp     interrupt_end

code    ends
        end
```

ワンラインNEWS
日立製作所は同社のスーパーパソコン「Sシリーズ」にUNIXの最新版「OSF/1」を搭載し，受注活動を開始した．OSF/1を実用化するのはこれがはじめて．

ます．ブロックデバイスのときにはこのほかに，情報に関する値を設定します．それからinterrupt_endを読んで処理を終了します．

XSCSI.ASM

次に，SCSI処理部分であるXSCSI.ASMについて説明しましょう．このデバイスは，IOCTL以外の処理はまったくしません．ioctl_inでは，テーブルentry_tableのアドレスを返すだけです．ほかの処理が呼ばれると，すべて命令コードエラーとなって返ります．デバイスドライバの形式を取っていて，かつ通信方法もデバイスドライバの方法を用いますが，その本質は単なるTSRといってもかまいません．

entry_tableは，読み書き，デバイスの容量取得，SCSIコマンドの発行各処理ルーチンのアドレスが記憶されます．また，SCSIでエラーが発生した場合の状態を示すバッファポインタのアドレスもここに記録します．

ブロックデバイスをサポートしたい場合，基本的には読み書きのルーチンと，容量がわかればアクセスできます．ですから，この部分にコンパチビリティを取っておけば，たとえばディスクBIOSを経由してアクセスする，98標準のディスク構造ではないディスクを扱う，というようなことも可能となります．

さて，読み書きルーチンでは，読み書きを発行するコマンドパケットを作成し，それをscsi_primeに渡します．scsi_primeはfarのエントリですから，csを先にプッシュし，farcallをエミュレートしています．

ここで，読み書きされるデータ数はブロック数からバイト数に変換されます．SCSIデバイスというのは1ブロック512バイトが標準ですから，そのように変換しています．98においても，物理的なセクタ長は512バイトであり，それをまとめて論理的に1KBや2KBなどのセクタとして扱っているのです．

リスト2 XSCSI.ASM

```
;--------------------------
; xscsi: simple scsi handler
;--------------------------

NECIF   equ     1
;ALWAYS  equ     1

ioctl_in:
        les     bx,packet
        les     bx,es:[bx].trans
        mov     word ptr es:[bx],offset entry_table
        mov     word ptr es:[bx+2],cs
        mov     ax,100h
        jmp     interrupt_end

error_command:
media_check:
build_bpb:
read:
nd_read:
read_status:
read_flush:
write:
write_verify:
write_status:
write_flush:
ioctl_out:
device_open:
device_close:
removable_media:
output_until_busy:
deinstall:
        mov     ax,8103h
        jmp     interrupt_end

; table for entry
entry_table     label   word
        dw      read_write
        dw      get_capa
        dw      scsi_prime
        dw      sense_buffer

sense_buffer    db      40 dup (?)

;--------------------------
; read/write routine
;--------------------------
; input
;       es:di data buffer
;       cx    sector count
;       al    lun/scsi id, or drive number.
;       ah    r(0)/w(1) mode
;       dx:bx address
; output
;       al      status
;       cx      transfered sector count
;       es:di   advanced data buffer

scsi_packet     db      10 dup (?)

read_write      proc    far
        push    si
        push    ds
        push    cs
        pop     ds
        mov     scsi_packet,8
        test    ah,1
        @if     nz
                mov     scsi_packet,10
        @endif
        mov     scsi_packet+1,dl
        mov     word ptr scsi_packet+2,bx
        mov     scsi_packet+4,cl
        mov     scsi_packet+5,0
        push    ax
        and     al,not 7
        or      scsi_packet+1,al
        pop     ax
        and     al,7

        ; block -> byte
        mov     ch,cl
        mov     cl,0
        shl     cx,1

        mov     si,offset scsi_packet
        push    cs
        call    near ptr scsi_prime

        ; byte -> block
        mov     cl,ch
        mov     ch,0
        shr     cx,1

        ; check condition


        pop     ds
        pop     si
        ret
read_write      endp

;--------------------------
; get capacity routine
;--------------------------
; input
; al    lun/scsi id, or drive number
;
; output
; al    status
; dx:bx sector number
;
get_capa        proc    far
        push    cx
        push    si
        push    di
        push    ds
        push    es

        ; create packet
        mov     scsi_packet,25h
        push    ax
        and     al,not 7
        mov     scsi_packet+1,al
        pop     ax
```


ワンラインNEWS
日本電気はRISC MPU搭載のデスクトップ型ワークステーション2機種を発売した．下位機はシステム価格88万円で，RISC型ワークステーションでは最低価格を実現．

その次のget_capaもほぼ同様に，容量取得のコマンドパケットを作成し，それを引数に取りscsi_primeを呼び出すだけです．

scsi_primeは，基本的にはその下請けで，ここではNECSCSIで定義されている_scsi_primeを呼び出します．ここでなんらかのエラーが発生した場合には，sense_dataのコマンドを発行します．結果がreset conditionだった場合には，もう正常にデバイスはアクセス可能ですから，もう1度同じコマンドを発行します．そうでない場合には，そのまま返ります．

そのあとに，NECSCSIをインクルードします．この中でend_labelが定義されます．XSCSIにおける初期化ルーチンex_initは，NECSCSIの中で定義されているhard_initを呼び出すだけで，実際には何もしません．

◆NECSCSI.INC

さて，では最後にハードウェアの部分を担当するNECSCSIを見てみましょう．最初に，インストール時に出力するメッセージが定義してあります．それから，ワークエリアがあり，message_inというルーチンがあります．

このmessage_inというのは，98のSCSIボードで使用されているWDの石に特有な処理をするためのものです．BIOSを呼び出すときには，(最大)転送データ数を渡します．これがBIOSの中でチップに設定されるわけですが，なんらかの障害などで指定したデータが転送終了しない場合，石がハングアップ状態になります．これを回避して処理を継続するためのものが，message_inというルーチンなのです．このルーチンでは，SCSIのシーケンスの最後の部分，ステータスフェーズなどの処理をして終了します．SCSIの処理自体は，SCSI-BIOSが利用できるので非常に簡単にすんでいます．

その次にある_scsi_primeが，SCSIコマンドを発行し，メッセージの転送を行うメインルーチンです．渡されたSCSIコマンドバッファをローカルデータにコピーし，それに98のBIOSで

```
        and     al,7
        sub     dx,dx
        mov     word ptr scsi_packet+2,dx
        mov     word ptr scsi_packet+4,dx
        mov     word ptr scsi_packet+6,dx
        mov     word ptr scsi_packet+8,dx
        mov     cx,8
        mov     si,offset scsi_packet
        mov     di,si
        push    cs
        push    cs
        pop     ds
        pop     es
        push    cs
        call    near ptr scsi_prime

        ; receive result
        mov     dx,word ptr scsi_packet
        mov     bx,word ptr scsi_packet+2

        pop     es
        pop     ds
        pop     di
        pop     si
        pop     cx
        ret
get_capa        endp

;----------------------------
; scsi prime routine
;----------------------------
; input
; ds:si command buffer
; es:di data buffer
; cx    data transfer length
; al    scsi id
; ah    r(0)/w(1) mode.
;
; output
; ah:al key:status
; cx    transfered length
; es:di advanced pointer
;

ax_save dw      ?
cx_save dw      ?
sense_data_command      db      3,0,0,0,0,0

scsi_prime      proc    far
        mov     cs:ax_save,ax
        mov     cs:cx_save,cx
        call    _scsi_prime
        mov     ah,0

        @mc
                cmp     al,2
                @cond   ae
                cmp     al,6
                @cond   b
        @if
                ; sense
                push    ax
                push    cx
                push    si
                push    di
                push    ds
                push    es

                push    cs
                pop     ds
                push    cs
                pop     ds
                mov     si,offset sense_data_command
                mov     di,offset sense_buffer
                mov     cx,40
                mov     ax,ax_save
                mov     ah,0
                call    _scsi_prime

                ; check reset condition
                mov     al,sense_buffer+2
                and     al,0fh
                cmp     al,6

                ; restore registers
                pop     es
                pop     ds
                pop     di
                pop     si
                pop     cx
                pop     ax
                mov     ah,cs:sense_buffer+2

                ; jump
                @if     z
                        mov     ax,cs:ax_save
                        mov     cx,cs:cx_save
                        jmp     near ptr scsi_prime
                @endif
        @endif

        ret
scsi_prime      endp


;----------------------------
; hardware initialize routine
;----------------------------
; input
;       none
; output
;       none

ifdef NECIF
        include necscsi.inc
elseifdef ALWAYS
        include always.inc
endif

;----------------------------
; initialize part
;----------------------------
ex_init proc
        call    hard_init
        ret
ex_init endp
```

使用されるヘッダを付加します．それからコンポジットタイプのBIOSコマンドを呼び出します．次にステータスをチェックし，これが2bhであれば，データ長が不正であるということで，実際にはシーケンスが中断されてしまっています．ですから，先のmessage_inを呼び出し，事後調節をします．そして復帰するわけです．

この部分までが常駐時に必要なルーチンであり，そのあとにend_labelを定義して非常駐域を指定した後，hard_initを定義しています．ここでは，hard_initも何もせずに返るルーチンになっています．

98のSCSI-BIOSの仕様については，コーラルから出版されている，SCSI関係の書籍を参照してください．

プログラムの作成

tasmでは直接SYSファイルを作成

さて，以上のファイルからデバイスドライバを作成するためには，次のようなシーケンスを実行します．

```
A:¥>tasm /DPRIME _DEV dev-std,xscsi
A:¥>tlink /t xscsi,xscsi.sys
```

masmとlinkを使用する場合には，1度exeファイルを出力してからexe2binなどでsysに変換することになります．tlinkは直接com形式のファイルを出力できるので，こちらを使えば必ずしも変換する必要はありません．

PRIME_DEVを定義することによって，今回必要な部分のみが生成されます．こうしてできあがったXSCSI.SYSは，CONFIG.SYSの中で定義するか，あるいはADDDRVなどで付加することによってインストールすることができます．

次回以降に掲載するドライバは，このxscsiの存在を前提としているので，それらに先立ってxscsiはロードされなければなりません．

リスト3　NECSCSI.INC

```
;--------------------------
; NEC scsi interface handler
;--------------------------

scsi_mes        db      'ExSCSI handler for NEC SCSI BIOS',0dh,0ah,'$'

myparam db      16 dup (?)
trans_len       dw      ?
trans_id        db      ?

message_in      proc    near
        ; save register
        push    ax
        push    bx
        push    cx
        push    es
        push    cs
        pop     es

        ; status phase
        mov     ah,1bh
        mov     al,trans_id
        mov     bx,dx
        inc     bx
        mov     cx,1
        int     1bh

        ; message phase
        mov     ah,1fh
        mov     al,trans_id
        mov     bx,dx
        inc     bx
        mov     cx,1
        int     1bh

        ; negate ack
        mov     ah,3
        mov     al,trans_id
        int     1bh

        pop     es
        pop     cx
        pop     bx
        pop     ax

        ret
message_in      endp

_scsi_prime     proc    near
        push    bx
        push    dx
        push    si
        push    ds
; first copy contents of command buffer
        push    es
        push    cs
        pop     es
        mov     bx,di
        mov     di,offset myparam+4
        movsw
        movsw
        movsw
        movsw
        movsw
        movsw
        push    cs
        pop     ds
        pop     es
        mov     dx,offset myparam
; set parameter of cdb
        mov     myparam+1,0

        or      cx,cx
        @if     nz
                mov     myparam+1,48h
                or      ah,ah
                @if     z
                        mov     myparam+1,44h
                @endif
        @endif

        mov     ah,al
        shr     ah,1
        shr     ah,1
        shr     ah,1
        shr     ah,1
        shr     ah,1
        and     ah,7
        mov     myparam,ah

        mov     myparam+2,10
        cmp     myparam+4,20h
        @if     b
                mov     myparam+2,6
        @endif
; other register set
        mov     ah,9
        and     al,7
        add     al,0c0h
        mov     trans_id,al

        mov     trans_len,cx
        int     1bh
        sub     trans_len,cx
        mov     cx,trans_len

; check data length error
        cmp     ah,2bh
        @if     z
                call    message_in
        @endif
        mov     di,bx

; return
        mov     al,myparam

        pop     ds
        pop     si
        pop     dx
        pop     bx
        ret
_scsi_prime     endp

; from here, not resident part
end_label:

hard_init       proc    near
        ret
hard_init       endp
```


ワンラインNEWS
OADG(PCオープン・アーキテクチャ推進協議会)はシンボルとなるロゴマークを発表した．また新会員としてセガエンタープライズなど3社が加わった．

情報処理技術者試験
第2種合格！トレーニング
第10回 CASL

情岡情報ビジネス専門学校　加藤忠宏・藤波正

出題傾向と対策

CASLの問題は，午後の部で3問出題されます．昭和62年に，CAP-XからCASLに変更されてからの出題傾向は表1のとおりですが，年々難しくなっていくようです（一般に，1種の方がやさしいようです）．

出題の内容から傾向を分析してみますと，文字列操作に関するもの，ビット操作に関するもの，四則演算に関するもの，配列に関するものと4種類に分けられます．なかでも，文字列操作に関するものの出題頻度がもっとも高くなっています．

アセンブラ言語を苦手とする人も多いようですが，まずは言語の仕様を十分に理解することがたいせつです．また，入力と出力を明確にしておき，ある程度アルゴリズムの推測をしてからプログラムを追うことも，CASLを攻略する手だと思います．

各分野別の対策

■配列に関する問題

いろいろな問題が出題されていますが，処理対象のデータ構造が配列を持つ問題です．基本的に，配列の先頭アドレス（記号アドレス）をそのまま使うか，レジスタに持たせ，ポインタ用のレジスタを更新させて，データを取得するやり方です．

●表1　CASLの出題傾向

年度	春の出題	秋の出題
昭和62年	チェックディジットの算出 利用者識別コードのチェック シフトによる除算	ゼロ抑制と表示 16進数字を2進数へ 競技者の得点の計算
昭和63年	最大値・最小値の算出 試験問題の選択人数調査 年月日データの圧縮	入力数字列の編集 10進数字を2進数へ ビットカウント
平成元年	数値の正負 指定データの検索 演算処理	試験の採点プログラム 除算のプログラム 検査文字作成プログラム
平成2年	文字列比較プログラム 電話機管理プログラム 文字列圧縮プログラム	文字列置換プログラム 文字列置換プログラム ビット転送プログラム
平成3年	同一文字の検索 バッファ操作 倍精度整数の計算	

●表2　代表的な問題

文字列操作に関する出題	ゼロ抑制 データチェック	文字0を間隔文字に置換 同一文字の検索 チェックディジットの作成 文字列の置換 文字列の圧縮
ビット操作に関する出題	ビットカウント ビット位置検索 ビット列の移動	領域内のONビットの数を数える 領域内のON/OFFビットの位置を検索 ある領域内のビットデータを別の領域に転送
四則演算に関する出題	掛け算（乗算） 割算（除算） 進数変換	
配列に関する出題	色々な問題	

●配列に関する問題のコーディング例

```
LABEL   LD    GR0,ARRAY,GR1 ……… 配列ARRAYからGR1番目の内容を取得
          処理 ……………………… 処理の実行
        LEA   GR1,1,GR1 ………… GR1の更新(配列の添字を作る)
        CPL   GR1,END ………… 配列の最後の要素まで処理したかを確認
        JMI   LABEL ……………… 全要素を処理したら終了．それ以外はLABELに
                                 ジャンプ
```

■ビット操作に関する問題

1語内のONビットの数を数えるには，左右に1ビットずつシフトして1語内のデータが，ゼロになるまでシフトを繰り返す方法が一般的です．ただし，右シフトの場合，最上位ビットがシフトした箇所に埋め込まれる命令がありますので注意してください．

また，1語内のON／OFFビットの位置を検索するには，論理演算命令を使用すると便利な場合が多いです．

■四則演算に関する問題

CASLには乗算・除算命令がないため，加算・減算命令を繰り返し実行して実現します．

また，進数変換は次の手順で実行します．

・10進から2進へ

10進数は文字として与えられます．そのため，文字コードから30H減算することで変換できます．

・16進から2進へ

0から9の文字は，文字コードから30H減算します．AからFは，文字コードから37H減算することで変換できます．

・2進から10進へ

10で除算して，文字コードから30H加算して変換できます．

・2進から16進へ

16で除算して変換できます．

■文字列操作に関する問題

ゼロ抑制，同一文字の検索，文字列の置換は配列操作の問題ともとれます．文字列（文字配列内）のデータと別のデータとの突き合わせであると考えればわかりやすいと思います．

チェックディジット・圧縮に関しては，過去の問題を再度，十分に検討して実力をつけるのが得策です．

●ビット列に関する問題のコーディング例(1)

```
                 :
LABEL1   JZE     LABEL2………… GR0が0でないならONビットあり，0ならLABEL2へ
         LEA     GR3,1,GR3…… ビットカウンタの更新
         SLL     GR0,1……… レジスタデータを左シフト
         JMP     LABEL1……… LABEL1へ無条件ジャンプ
LABEL2
                 :
```

●ビット列に関する問題のコーディング例(2)

```
                 :
         LD      GR3,ARRAY,GR1…… 配列ARRAY内のGR1番目のデータをGR3へ転送
         AND     GR3,ARRAY,GR2…… GR3とARRAY内のGR2番目を論理演算
         JZE     END……………… 同じ位置がONなら非0，OFFなら0となり終了
                 :
```

●乗算の場合のコーディング例（10倍する場合）

```
                 :
ST       GR1,WK……… 作業領域にレジスタの内容を設定
SLL      GR1,2……… GR1にGR1を4倍したものを設定
ADD      GR1,WK……… さらに元のGR1の値を加算してGR1に設定（元の値の5倍）
SLL      GR1,1……… GR1を2倍（元の値の10倍）
ST       GR1,WK……… GR1を作業領域に設定
                 :
```

●除算の場合のコーディング例

```
                 :
         LEA     GR1,0……………… 商の領域を0クリア
LABEL1   CPA     GR2,0,GR3………… 除数が被除数より大きいか
         JMI     END………………… 除数の方が小さければ終了
         SUB     GR2,0,GR3………… 除数から被除数を減算
         LEA     GR1,1,GR1………… 商を設定
         JMP     LABEL1…………… LABEL1へ無条件ジャンプ
END
                 :
```

EXAMPLE1

アセンブラ言語CASLが使用できる計算機COMETがある．その仕様とプログラムの説明およびプログラムを読んで設問に答えよ．

[プログラムの説明]

パリティビットを付加する副プログラムである．

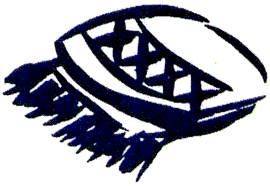
(1)パリティビットを付加したいデータがGR1に格納されてこの副プログラムに引渡され，偶数パリティをGR1に付加した後，主プログラムに戻る．

(2)主プログラムからGR1を経て副プログラムに引渡されるデータは，GR1の第1ビットから第15ビットまでの15ビットにビット列として設定されている．

(3)パリティビットは第0ビットに設定する．

[設問]

次のCASLプログラム中の□を埋めてプログラムを完成させなさい．なお，ラベルは手続き上必要でない限りつけないこと．



●リスト１

ラベル	命令コード	オペランド	説明
PARITY	START		
	ST	GR1, WORK	
(1)			レジスタの初期化
	SLL	GR1, 0	終了判定
	JZE	LAST	
LEVEL1	SLL	GR1, 1	ビットの抽出
(2)			1が抽出できたか
	LEA	GR2, 1, GR2	1のビット数のインクリメント
	JMP	LEVEL1	
LEVEL2	JZE	LEVEL3	
	JMP	LEVEL1	
(3)			偶数か奇数か
	JZE	LAST	
	LD	GR1, MASK2	マスクデータのロード
LAST	(4)		パリティの付加
	RET		
WORK	DS	1	作業域
MASK1	DC	#0001	奇偶検査ビット
MASK2	DC	#8000	パリティビット
	END		

解　説

・ビット操作のテクニック

本問の解答の作成に際して利用したビット操作のテクニックは次のとおりです．

①０回シフトによるゼロ判定
②論理左シフトを用いた１のビットの抽出
③排他的論理和を用いたレジスタの正負判定
④論理和を用いたパリティビットの付加

・プログラムの解説

GR1に16進数2000を格納し副プログラムに引渡しトレース結果を行ってみると図１のようになります．16進数2000中の１のビットは１個ですから偶数パリティを付加する必要があります．

・トレースのポイント

以下にトレースの順とそのポイントを示します．

(1)説明欄の示唆からレジスタの初期化であることがわかります．CASLのプログラムではループ前のレジスタの初期化は指標もしくは総和の初期化であることが多く，本プログラムの総和はGR2でカウントされていることからGR2に０を格納します．

(2)論理シフトのループを作ることによって第０ビットに，第１ビット～第15ビットの各ビットを順次抽出していきます．抽出したビットが１のときはGR2のカウント，０の場合は終了判定を行うことになるので，(2)は正分岐となります．

(3)未使用のラベルがあることからまずLEVEL3を記述します．次にデータ定義域に未使用のマスクデータMASK2があることからこれを用いた論理演算であることがわかります．目的の数値が偶数であるか奇数であるかを判定するためには第15ビットを検査すればよいのです．目的の第15ビット以外はマスクしたいのでANDを用いることになります．

(4)ビットの付加にOR命令を用いるのはアセンブリ言語の常識です．別解として

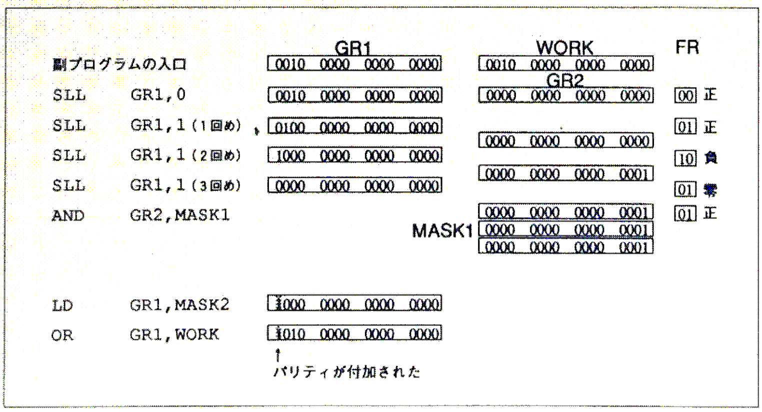
［ADD　　GR1, WORK］

を記述してもプログラムは成立するのでおぼえておいてください．

解　答

(1)		LEA	GR2, 0
(2)		JPZ	LEVEL2
(3)	LEVEL3	AND	GR2, MASK1
(4)		OR	GR1, WORK

●図１



EXAMPLE2

アセンブラ言語CASLが使用できる計算機COMETがある．その仕様とプログラムの説明およびプログラムを読んで設問に答えよ．

[プログラムの説明]

二分探索を行う副プログラムである．

(1)探索したいキーを1文字がGR0に格納されてこの副プログラムに引渡され，二分探索法による探索を行い，探索結果に応じた出力を行った後，主プログラムにもどる．

(2)探索の対象となる配列TABLEはすでに昇順に整列されているものとする．

(3)二分探索の基本原理は図2のとおり．

(4)探索すべき配列TABLEは50文字格納している．

[設問]

次のCASLプログラムの［　］を埋めてプログラムを完成させなさい．なお，ラベルは手続き上必要でない限りつけないこと．

●リスト2

ラベル	命令コード	オペランド	説明
BINARY	START		
	ST	GR0, KEY	キーの退避
	LEA	GR1, 0	配列の始端を MIN に
	ST	GR1, MIN	
	(1)		配列の終端を GR2に
	ST	GR2, MAX	
LOOP	(2)		探索位置の決定
	ADD	GR1, MIN	
	ADD	GR1, MAX	
(3)			1/2を行う
	LD	GR0, TABLE, GR1	
	CPL	GR0, KEY	配列データとキーの比較
(4)			
	JMI	LABEL3	
	LEA	GR1, -1, GR1	終端の更新
(5)			
POINT	LD	GR0, MAX	終了判定
	CPA	GR0, MIN	
	(6)		
	JMP	LOOP	
LABEL1	OUT	OKMES, OLENG	キーの照合処理
	JMP	LAST	
LABEL2	OUT	NOMES, OLENG	キーの未照合処理
LAST	LD	GR0, KEY	レジスタの復元
	RET		主プログラムへの復帰
LABEL3	(7)		始端の更新
	ST	GR1, MIN	
	JMP	POINT	
KEY	DS	1	照合キーの退避域
MAX	DS	1	配列の終端
MIN	DS	1	配列の始端
TABLE	DS	50	配列
OKMES	DC	'OK MACHING'	照合メッセージ
NOMES	DC	'IT NATHING'	未照合メッセージ
OLENG	DC	10	出力文字数
	END		

解　説

以下にトレースの順とそのポイントを示します．

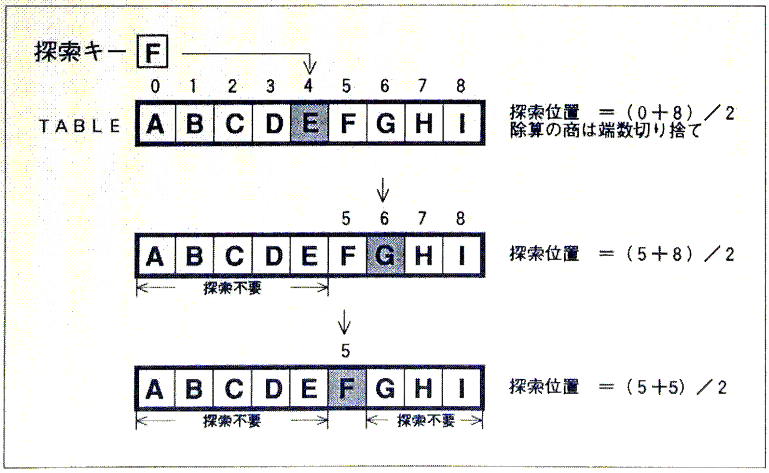
(1)空欄直前の命令構成と説明欄を参照すれば容易に解答できるでしょう．配列数は50ですが，末端の変位は49であることから49をGR2に設定します．

(2) GR1の機能を切り換えているため，回答者は小考を要します．手がかりは空欄(3)直後のLD命令でGR1が指標として利用されているところです．解説にもあるようにこのブロックでは探索キーと比較を行う配列データの指標をGR1に求めていること，この命令を反復利用することからGR1の初期化を行うことが適当であると推測されます．

(7)配列データがKEYより小さい場合の処理です．図と説明から配列の始端のインクリメントであることがわかります．

(6)二分探索法を用いたデータ探索の終了判定はMINとMAXの比較です．探索キーに対応するデータが配列中に存在しない場合には必ずMIN>MAXとなります．分岐命令の飛び先は未照合処理を行うLABEL2がよいでしょう．

●図2



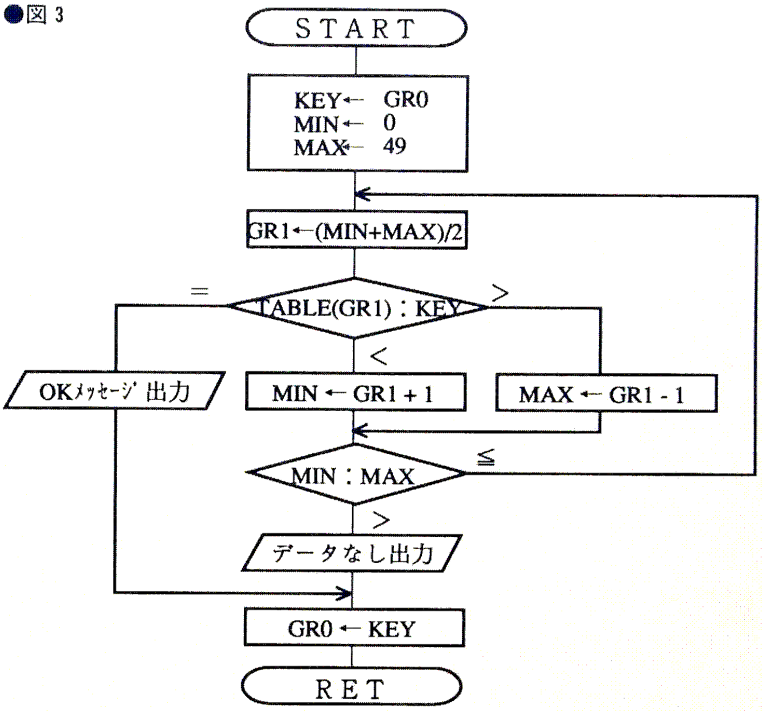
(4)キーの照合が行われた場合の処理を実行するためにLABEL1への分岐を行います．

(5)前後の関係から配列データがKEYより大きい場合の処理であることがわかります．図と説明からMAXのデクリメントであることがわかります．

解　答

(1)	LEA	GR2，49
(2)	LEA	GR1，0
(3)	SRA	GR1，1
(4)	JZE	LABEL1
(5)	ST	GR1，MAX
(6)	JMI	LABEL2
(7)	LEA	GR1，1，GR1

●図 3



●図 4

●ポイント１　ループ前の空欄はループ中にヒントあり

```
          [   (1)   ]  ------------┐
               :              初期値が設定されているか？
 ┌─> LOOP
 │        LD   GR0,TBL,GR1 <------┘
 │             :
 └──────  JMP  LOOP
```

●ポイント２　比較演算命令の直下は分岐命令

```
  CPA     GR1,KEY
  [   (2)   ]        比較と分岐はセット
                     ∴　分岐命令
```

●ポイント３　分岐命令の直前は必ずしも比較演算命令ではない

```
  [   (3)   ]        フラグのセット
  比較分岐命令   LABEL1
```

●ポイント４　プログラムの構造を考えよ

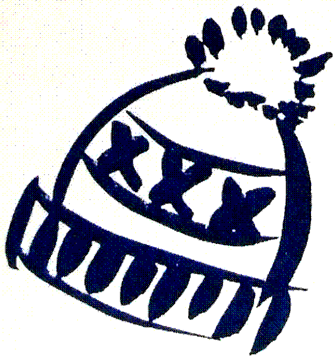
一般にCASLのプログラムは以下の構造を取る場合が多い
図で示したポイントが空欄として出題されることも多いので注意したい

```
 ┌──> LOOP1    外側の終了判定 ────┐
 │             LOOP2の準備        │
 │ ┌─> LOOP2   内側の終了判定     │
 │ │               内側の処理     │
 │ └────────   JMP     LOOP2      │
 │             終了判定の準備     │
 └──────────   JMP     LOOP1      │
       LAST    EXIT  <────────────┘
```

●ポイント５　説明欄を参照して解答せよ

説明欄を参照すれば、トレースしなくても解答できる場合もある
(例)

マスクをかける	→ ＡＮＤ命令	単純な除算	→ ＳＲＡもしくはＳＲＬ命令
ビットの反転	→ ＥＯＲ命令	余りを求める除算	→ ＳＵＢ命令
ビットの付加	→ ＯＲ命令	初期化	→ ＬＥＡ命令



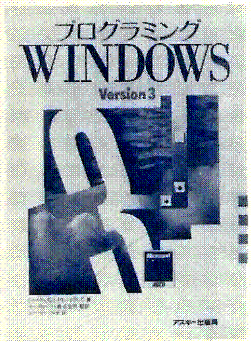
Book Review

■中村正三郎

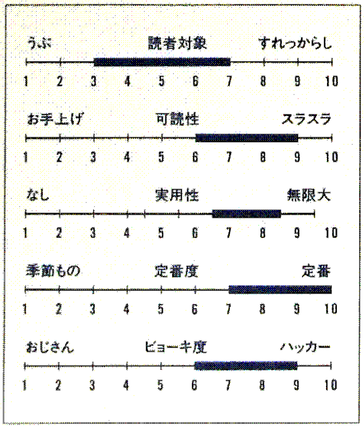
Windowsプログラミングをはじめるなら真っ先に読むべき教科書

■プログラミング WINDOWS Version 3

Charles Petzold著
マイクロソフト監訳
エー・ピー・ラボ訳
アスキー出版局　7,800円



■評価チャート



著者Charles Petzoldは，Windowsプログラミングの世界では大変に著名な人物である．そしてまた，OS/2 PMプログラミングでも著名である．WindowsとPM，それぞれのプログラミングの著書がベストセラーになったからだ．とりわけWindowsの本は人気が高い．またそれにふさわしい内容を備えている．彼は，これらGUI (Graphical User Interface) プログラミングのオーソリティといえる．

「Windowsで，こうやりたいときは，どうすればいいの？」「あ，それならペゾルド本に書いてあるよ」

このように彼の本は，プログラマの間では通称「ペゾルド本」と呼ばれ，バイブルになっている．プログラマは彼の著書を信頼している．マイクロソフトのSDK (開発キット) のマニュアルよりもずっと信頼している．SDKのマニュアルは，怪しげな部分が散見される．翻訳段階で間違ったらしい怪しげさ，あるいは英文マニュアルからして間違っているらしい怪しげさ，それを訂正せず，そのまま訳したらしい怪しげさが，ちらついている．

前置きが長くなった．本書は，Programming Windows 2nd Editionの邦訳である．原書の分厚さを反映して1000ページを超える大変なヴォリュームである．前作，Programming Windowsの邦訳は，同じくアスキーから出ており，これは上下2分冊になっていて，その点使いやすかった．前作なら通勤電車の中で読むこともできたが，今回の厚さでは不可能だろう．1冊にまとめよという要望があったのだろうか．

内容面で，前作とどの程度違うかといえば，大筋において大した違いはない．前作がWindows 2用であったのに対し，その後発表されたWindows 3用に書き直したものが本書だからである．目立つところでは，DDEとMDIの章が追加された．これ以外は，サンプルプログラムも解説も，ほぼ同じと思ってよい．もちろん，Windows 3用に細かな点で修正が入っているし，(訳者によれば)訳はよりわかりやすくなったという違いはある．

内容はツボを的確に押さえており，心憎いほどだ．サンプルプログラムは，単なるサンプルに終わらず，そのままちょっとした道具として役に立つものや，改良を加えれば道具に仕上げられそうなものになっている．各章の解説もテンポよく，Windowsプログラミングのさまざまな側面に，小気味よい説明を行っている．SDKの無味乾燥なマニュアルでは，理解不能だったことが，よくわかるようになる．なるほど．あれは，こういう意味であり，このAPIはこう使うものだったのかと．

本書全体の構成に評者は感心した．流れが大変によい．SDKのマニュアルにあるAPIのジグソーの断片が，ひとつの絵にまとまる様を見る思いである．各章，各節が次にうまくつながっているので，興味を失わず読み進めることができる．もっとも，一気に本書を読み上げるのは大変な作業ではあるが．

ともあれ，これからWindowsのプログラミングに取り組もうと思う人は，まず本書を読むところから手をつけてみるのが近道だ．最近はVisual BASICのように，Windowsの複雑さを巧妙に隠している言語や開発ツールもぞくぞく登場している．しかし，そ

ワンラインNEWS
カシオ計算機はアウトラインフォントを搭載したページプリンタ「ページプレストCP-2150」を発売した．これは同社初のアウトラインフォント搭載ページプリンタ．

れらを使う場合でも，本書で得た知識は大いに役に立つだろう．ましてやオーソドックスにC言語で書こうと思っているのなら，真っ先に読み，サンプルプログラムを試した方がいい．

実によく書けているWindowsプログラミングの教科書である．

最後にもうひとつ，負けず劣らずよく書けているWindowsプログラミングの教科書を紹介しておこう．ピーター・ノートン，ポール・ヤオ著，富士ソフトウエア教育出版部訳『Windows 3.0 パワープログラミング上下』(富士ソフトウエア)である．ペゾルト本には載っていないことも載っており，この2冊は互いに補完しあっている．両方を手元においておくとよいだろう．

91年度洋書ベスト10決定！「Compilers」が第1位に

古田島義和（神田・書泉グランデ）

91年を総決算！　今回の洋書のベスト10は，91年の1年間に売り上げた図書を120位までリストアップし，その中からベスト10を掲載しています．ご覧のようにAddison Wesley（以下AW）の「Compilers」(Aho著）が第1位で毎年上位に登場している常連です．これはAWのハードカバー版を日貿出版がペーパー版にして発売しているもので，定価もハードカバーの約3分の1の安さで，Compilerを学ぶユーザー必読書であり，経済面でも大いに助かっているところが，実績に現れているのでしょう．

余談ですが，Prentice Hall(以下PH)のKernighan & Ritchie共著「The C Programming Language 2/E」（訳本は共立出版の「プログラミング言語C 2/E」）のアンサーブック「The C Answer Book 2/E」(Tondo & Gimpel共著，訳本は啓学出版の「Cアンサーブック2/E」）が同社から，AWからは，「Design & Implementation of 4.3 BSD UNIX Operating Systems」（訳本は丸善の「UNIX 4.3BSDの設計と実装」）のアンサーブック「Design & Implementation of 4.3 BSD UNIX Operating Systems Answer Book」(訳本はトッパンの「UNIX 4.3 BSDの設計と実装アンサーブック」）が出版され，いずれもベストセラーになっています．こうしたアンサーブックのなかで今回，Compilerのアンサーブックともいえる作品が，PHから「The Art of Compiler Design」(Pittman著）として発売予定です．

120位全体を見ても，C言語，UNIX，UNIX Networkといったところがよく売れ，不動の地位を確立しているようです．そして，今年目立ったのは，Macintoshで，「System 7」にバージョンアップされ，春に幕張メッセで開催されたMacWorld Expo/Tokyo '91以来，「ResEdit Reference. Complete」を中心に，既刊のAW「Inside Macintosh Series I～V」の第6巻目もSystem 7の発表に合わせて出版され，好調な売れ行きをしめしています．そのほか，定番として，M. I. T Pressの「Mac Scheme: Manual & Software」がロングセラーで，第15位にランキングされています．バージョンアップといえば，DOSもVer. 5. 0になり，91年暮れまでには，20数冊以上の出版点数を数え，なかでもMicrosoft Press（以下MS-Press）の「MS-DOS Programmer's Reference」やAWの「Undocumented DOS Ver. 5. 0」などが好調な売れ行きです．Windows関係では，MS-Pressの「Programming Windows 2/E」の訳本が早くもアスキーから出版され，今回の和書ベスト10の第9位で健闘しています．

ほかに話題の分野は，オブジェクト指向でしょう．「C++ Programming Language 2/E」(Stroustrap著）をはじめ，同氏の「Annotated C++ Reference Manual」，「C++ Primer 2/E」(Lippman著)，「Object-Oriented Design」(Booch著)，「Objective-C」(Pinson & Wienor共著）など，この分野は，AW一色，といったところでしたが，後半からPHから，「Object-Oriented Modeling and Design」(Rumbaugh著)，「Object-Oriented Analysis 2/E」(Coad著)，同氏の「Object-Oriented Design」などが出版され，92年はじめからも続々と出版が予定されています．出版案内は，次号で紹介しますので，お楽しみに．

■和書&洋書ベスト10

（書泉グランデ調べ：1991年10月1日～10月31日，洋書は1991年度分）

1	スーパーエンジニアへの道	共立出版	1	Compilers	Addison Wesley
2	続・健康ソフトハウス物語	星雲社	2	Programming Perl	O'Reilly
3	NIFTY-Serve'91年版	星雲社	3	Undocumented DOS Ver.5.0	Addison Wesley
4	カッコウはコンピュータに卵を産む（上）	草思社	4	ResEdit Reference 2/E	Addison Wesley
5	電脳騒乱節Vol.3	技術評論社	5	C++ Programming Language 2/E	Addison Wesley
6	カッコウはコンピュータに卵を産む（下）	草思社	6	ResEdit Complete	Addison Wesley
7	笑うコンピュータ	技術評論社	7	Simple Book:An Introduction to Management TCP/IP	Prentice Hall
8	標準MINIXハンドブック	アスキー	8	Programming Windows 2/E	Microsoft Press
9	プログラミングWINDOWS Version3	アスキー	9	UNIX Network Programming	Prentice Hall
10	フリーソフトウェア派宣言	翔泳社	10	Object-Oriented Modeling and Design	Prentice Hall

◆ ◆ 佐々木哲のニューヨーク ◆ ◆

N.Y. システム通信

マルチメディアが流行言葉

10月末にLas Vegasで開催されたComdexへ行ってきました．今年の流行言葉はマルチメディアです．でも，マルチメディアって何でしょう？　今のところ，どうやら音が出ればいいようです．動画はまだまだ高い．CD-ROMで音も含めたシステムが標準的に作れるようになってきました．そのため会場はSound Blasterのような音源ボードが多いこと多いこと．VGAカードと一体化した製品もATI，Headland（Video 7）はじめ数社から出ていました．

Windows NT，そしてOS/2

このComdexのひとつの焦点が新WindowsとOS/2 2.0です．新Widnowsといっても，3.1ではありません．これはもう広範囲のベータテストも佳境に入り，あとはいつ出るかだけが問題です．32ビットOSであるWindows NTなのです．マイクロソフトは，これをすでに強力に宣伝しています．マイクロソフトのブースだけでなく，インテル・ブースなど，会場のあちこちで動いておりました．コンパックの486マシンで386版を，そしてそれと並べてRISC WS上でRISC版を動かしていました．見せ方がうまいのは，既存のWindowsアプリケーションだけではなく，32ビットでリコンパイルしたExcelなども一緒に動かしていること．なかなか安定して動いているように見えます．一部へのSDKの配布もはじめ，着々と準備は進んでいるようです．

それに対するはIBMの意地，OS/2 2.0です．これは来年3月にフルセットでの発売が延びたとはいえ，2万人にも及ぶ大規模なベータテストが進んでいます．会場でも簡単なプレゼンテーションを受けた人に無料で配布している始末．とはいえ，会場でOS/2 2.0が見れたのはIBMのブースくらい，ほかの一般のブースではほとんど見かけません．これはOS/2 2.0にとってきびしい状況ですね．

とはいえマイクロソフト，いったいOS/2をどうしてしまうのでしょうね？　1年ほど前まではOS/2をすすめ，2,600ドルという高価なベータ版2.0 SDKも広く販売していました．ところが，最近の発表ではいっさいのOS/2サポートをIBMのOS/2 2.0出荷にあわせてやめるといいます．初期のころからOS/2の開発をしていたソフトハウスはひどい目にあっていますが，これでとどめを刺されたようなものです．一方，IBMも急速なOS/2の普及をはかるために，386SX版の新PS/1にOS/2 2.0をインストールして出荷すると発表しています．ホームユースにOS/2??．どうもこの争いには巻き込まれたくないものです．

399ドルのお手軽CD-ROM

タンディから発売されていた399ドルの格安，高速のCD-ROMですが，一般にも広くPinnacle Micro*[1]からも販売されるようになりました．内蔵タイプとポータブルタイプのものがどちらも399.95ドルです．キャディを省いて簡略化した設計だけでなく，複雑なSCSIを避け，独自のインターフェースをあえて使うことにより，インターフェースカード込みでこの値段，しかもほかより2倍以上速いデータ転送速度を実現しています（1.6MB/sec）．マイクロソフトのCD Extensionはいうに及ばず，新たなマルチメディアにも対応しています．

同じOEM元のCD-ROMを販売するタンディのブースでは，これを286CPUのPCにのせ，386/33MHzの機種には高価な日立のCD−ROMドライブをのせ，並べてくらべています．同じ動画をCD-ROMから読み込み，画面に表示するのですが，高級機の方では画面がつかえながら動くのにくらべ，こちらのほうは高速でスムーズに表示されていました．CD-ROMもハードウェア，ソフトウェアともにお手軽になってきました．

Modem! Modem! Modem!

つい数か月前，安価な9600bps V.32モデムの氾濫を報告したばかりですが，さらに事態は進んでいます．すでにPractical Periferalやインテルなどは，同価格帯で50%速い14400bpsのV.32bisモデムを発売しています．9600bpsのV.32モデムも値段の底が299ドルと決まったようですが，500ドル程度のものが続々発表されています．なかには東芝NotebookやSharp/TIの軽量286Note-

ワンラインNEWS
米IBMは人指し指だけでラップトップ型コンピュータなどのカーソル操作ができる指示スティックを開発した．マウスの使いにくい環境での作業に向く．

bookに内蔵するV.32モデムまでありました。

持ち運びを考えたものも増えています。ひとつおもしろい製品ではTrail Blazerで名の知れたTelebitからのQBlazer[*2]があります。コンパクトな5cm角ほどの立方体にV.32，MNP5及びV.42bisのモデムが内蔵されています。なかなかおもしろいデザインですね。9Vバッテリで5時間ほど動作し，通信できるとか。

V.32bis+V.42bis+Class 2 G3 FAX

Forvalのように独自の規格で21,600bpsという物理速度を実現しているモデムもありますが，やはり一般的にはCCITTで規定されたV.32bis，14,400bps全2重のものが広く使われます。

90年のComdexで，次のようなスペックのモデムが台湾のモデムメーカーZyXELから発表されていましたが，やっと発売になっています[*3]。内蔵，外部式と3種類のモデルがありますが，どれもV.32bis，V.32，V22 bis，V.22，BELL 212A，V.21，さらにはV.33（4線），V.29，V.27terまでサポートしてます。データ圧縮，訂正ではMNP5，V.42bisをサポート，さらにはG3 FAXをEIA Class 2のコマンドセットとともにサポート。要は14,400bpsのV.32bisまで，またFAXまで，これ1台あれば何でも通信できるわけです。これだけのモデムにいくら払いますか？　なんとIntroductory Specialで449ドルからです。

V.32bis+V.42bis+MNP 10+Class 1, Class 2 G3 V.17 FAX

実は上のZyXELのモデムは，既存のデータポンプチップなどを使わず，すべて自分達で開発していました。それでいてこの性能，価格を実現しているのです。たいした技術力です。

とはいえ，現在の低価格のモデムの普及にはRockwell[*4]に代表されるモデム用チップセットの開発，普及があります。ある程度の技術力がないメーカーでも高機能モデムを企画，製造できるわけです。実際これまでの低価格V.32モデムの普及の影にはRockwell 9696ACというチップがありました。

今回Rockwellはさらに進んだ製品，RC96AC-WとRC144AC-Wを発表しています。スペックを並べるだけでも大変なのですが，V.32bis，V.32，V.22bis，V.22，V.21，V.23，V.17（14,400bpsのFAX），V.29，V.27ter，V.21 Channel 2。データ訂正，圧縮機能としてはV.42bis，MNP 5に加え，携帯電話などでの移動体通信に要求されるMNP 10まで含んでいます。FAXのATコマンドセットもClass 1，2ともにサポートされています。

これを実際に使ったモデムが，また安価に出回ることが期待できます。500〜600ドルのモデムですべてのことができてしまう時代になりました。

高速ビデオボードの開発を容易にするS3

VGA＋GUIアクセラレータのS3を実際に使っています。これはいいですね。隣の欄で西川さんも書かれているように，386SXマシンで快適にWindowsが使えてしまいます。実はこのチップ，VGAとはいえ，レジスタ類を見ると内部の拡張部分はIBMの8514/Aに酷似しています。8514/Aはご存知のようにXGAの登場とともに公式には消えた規格ですが，ATI 8514/Ultraをはじめ，互換高機能ボードが広く入手可能になっています。おそらくはS3も8514/A互換をめざして開発されていたのでしょうが，VGAアクセラレータとして出てきたのでしょうね。コマンド類も8段のパイプライン処理を取り入れ高速に効率よく実行されます。XGAでもこれはやっていないようです。

S3が重要なひとつの理由は開発の容易さです。ボードメーカーはproduction kitをS3より購入します。これにはサンプルボード，スペックのほかにAMIの協力で書かれたBIOS，各種ドライバ類が含まれています。ボードメーカーはこれを参考に実際に独自にデザインし（あるいはそのままコピーして），生産，販売するだけです。すでに多くのメーカーから安価にボード類が発売されています。まさに8514/A，死して皮を残すです。

おわび

先月はお断りもなくお休みして申し訳ありませんでした。私事ですが長いアメリカ駐在を終え日本に戻ってきております。NY通信もこれでひとまず幕を引き，来月からは新たな企画でお目にかかりたいと思います。ご愛読ありがとうございました。

＊1　PCD-100 internal $399.95/PCD-101 portable $339.95
Pinnacle Micro phone (714) 727-3300
19 Technology Irvine CA 92718

＊2　QBlazer $799. Telebit Corporation
1315 Chesapeake Terrace, Sunnyvale, CA 94089
phone (408) 734-4333

＊3　U-1496（外部型表示パネル付）$650（$945），U-1496E（外部型表示パネル無）$469（$695）.U-1496B（内蔵型）$449（$675）括弧内定価，特別価格年内有効
449XEL Communication Corp. US distributor Zero One Networking
4920 E. La Palma Ave., Anaheim CA, 92807
phone (714) 693-0333. fax (714) 693-0705

＊4　Rockwell International Digital Communication Division
P.O. Box C, Newport Beach, CA 92658-8902
phone (714) 833-4655, fax (714) 833-6898

米クレイ・リサーチは同社のスーパーコンピュータ「CREY・Y-MP」シリーズの下位機種で，ミニスーパーコンピュータクラスの機能をもつ「Y-MP・EL」を発売した。

西川和久の

希望的観測

どうしてこんなに違うのか!?
海外のパソコン市場にあって国内にはないもの……

1 Windowsを高速化するには CPU負荷を軽くするビデオチップがあればよい

年末になってやっと，日本でもWindowsが立ち上がる気配を見せているが，米国ではすでに実用期を迎えている．しかしそれにともない，いろいろな欠点が指摘されだしているのも事実である．中でもディスプレイドライバのCPUパワー依存度は，深刻な問題になっている．ある資料によると，CUIにおいて画面まわりに使われるCPUパワーは平均的に見て10%～20%であるが，Windowsにおいては70%～80%も使われているとされている．WindowsはCPUパワーの無駄使いなのだ．これを解決すべくVGAベースにさまざまなチップセットが現れだした．

調べてみると気がつくのだが，ディスプレイドライバがもっている多くのAPIのうち，頻繁に呼ばれるのがExtTextOut ( ) とBitBlt ( ) の２つである．ExtTextOut ( )は，最終的に文字をビットマップに変換しBitBlt ( ) しているので，BitBlt ( )をハードウェア・アシストすれば，かなりの効果が期待できる．たとえば，SuperVGAであるET4000XXは，従来の機能プラスBitBlt ( ) だけをハードウェア化している．この機能を使ったWindows用ディスプレイドライバを組み込めば，ある程度CPU負荷を減らすことができるのだ．ちょと話はそれるが，IBMが押しているXGAも，このBitBltやPaint/Line系，バスマスタなどCAD/CAM，マルチメディアに適したハードウェアがふんだんに盛り込まれている．CPU依存度を低減化するという意味では，同じ効果を考えて設計されているのだ．

2 S3を使うと386マシンが486マシンより速くなってしまう

S3と呼ばれるチップセットでは，ディスプレイドライバに必要とされるかなりの部分をハードウェア化してしまった．私も簡単に評価したのだが，その結果は驚くべきものである．i80486／33MHz＋ET4000とi80386SX／20MHz＋S3の組み合わせでベンチマークテストを行ったところ，CPUパワーで圧倒的に負けているはずのi80386SX／20MHz＋S3のほうが，最低でも４倍，最高ではなんと24倍も速いのだ．Excel，PageMakerなど本格的なWindowsアプリケーションも試してみたが，断然速い．いかにこれまでCPUパワーを浪費していたかが目に見えてわかるのだ．このほかにも，WindowsのGDIを完全にハードウェア化したビデオカードなどが最近の雑誌を賑わしている．今後さらに拍車がかかることが予想される．

しかし日本に目を向けると，ハイレゾ化するビデオカードが若干出ている程度で，ここまでWindowsにターゲットを絞ったものは存在しない．パソコン本体がビデオまわりを固定して設計されているため，互換カードなど出る余地は全く残されていないのだ．ユーザーは，WindowsのビデオドライバがCPUパワーを無駄

ワンラインNEWS
ジャストシステムはオフィス向けのDTPシステム「大地オフィス・システム30」を発売した．また「大地オフィス・LANPACK」を来春発売する．

使いしているのを知りつつ，泣く泣くマシンをグレードアップするしか手段がない．ローカルアーキテクチャの悲しい定めなのだ…….

最後の手段はDOS/Vである．「IBM PC/ATとVGA」をベースにした日本語環境であれば，ドライバを日本語化さえすればS3など最新鋭の海外ハードウェアがそのまま使える．VGA，SuperVGA，XGAがそうであったようにS3も…….問題は誰が日本語ドライバを書くか!?である．(^^;

3 通信もFAXもできるFAXモデムが普及 国内では商売にならないのかな？

最近海外では，RS-232Cポートへ接続するFAXモデムがかなりの数出回っている．2400bpsのモデムが付いているのは当然のこと，MNP Class 5やV.42bisを搭載しているものまで，さまざまなバリエーションがあるのだ．肝心のFAXを制御する部分は，当初独自仕様のものが多かったが，ここ1年ほどのうちに，従来のATコマンドを拡張し「AT+Fxxxxx」で制御する「TR-29 Class 1(または2)」と呼ばれる規格に統一されだした．

カラクリは非常に単純で，モデムの設定を行った後，転送速度を19200bpsへもちあげ，MH圧縮形式の画像データをそのままRS-232Cポートへ送り出すようになっている．サイズも据置タイプからポケットタイプまでいろいろなタイプがあり，気になるお値段も日本円に直して20,000円から30,000円程度である．早い話，国内でポケットモデムが買える程度の値段でFAXまで付いてくるのだ．パソコン通信とFAXがモデム1台で手軽に扱えるため，海外ではかなり普及しだしている．

面白いことに日本語Windows上で米国版のWinFAX ProというWindows用ソフトウェアを使うと，FontProなどスケーラブルフォントを含め，日本語までもきれいに送信できてしまう．WinFAX Proは，Windowsアプリケーションから見るとプリンタドライバになっていて，印刷を開始すると電話帳が開き送信相手を選択した後，FAX送信が始まる仕掛になっているのだ．日本語Windowsでは，WIFEフォントに対応していないプリンタドライバ用にDISPATCH.BINと呼ばれるディスパッチ・ドライバが付属しているが，米国版のプリンタドライバへも適応できたというわけだ．FAXはもともと画像データなので，受信については全く問題はない．

これはなかなかおいしい話である．日本の商社がほっておくわけがない．DOS版のFAXソフトにしても，簡単な通信ソフトとMH変換する時に漢字を考慮すればよいだけの話であるから，それほど開発や移植が難しいとは思えない．しかし私の知っている限り，そのようなFAXモデムは現在国内では発売されていないのだ．理由はいろいろ考えられるが，もっとも大きな原因として「データを19200bpsで安定して送信する必要がある」ことだ．たとえば，CPUのパワー不足やフロッピーアクセスなど，送信中に少しでもデータが途切れる要因があると，相手であるFAX側が即異常終了してしまう．その上，国内標準機であるPC-98の8/16MHzクロック系マシンは，分周比の関係から9600bpsまでしかサポートされていない．かなりの割合で使えないマシンが多く存在するのだ．さらに国内トップシェア・ワープロ，一太郎Ver.4の文書ファイルフォーマットが公開されていないため，印刷イメージ通り送信できないなど，こんなことでは売っても仕方がない……これが真相ではないだろうか!?

4 日本の一般と世界の一般…… 文化の差までいくと難しい

「海外では一般的で，国内では全くない」，パソコンに関係するこのようなものは，今回の2つに限らずいろいろある．最近国内でもよく使われだしたのは，プリンタポートによるLANアダプタやSCSI/HDDアダプタである．J-3100，AX，DOS/Vなどで利用でき，需要もある程度見込めるため普及したと考えられる．

しかし，罫線などその逆のケースもけっこうある．パソコンの世界でも結局，文化的なレベルまで話がさかのぼることが多いため，そう簡単には結論がでない．世界的な一般をとるか．国内での一般をとるか．難しい問題である．

Computer Laws

コンピュータの法律

他人にプログラムを作成させた場合の著作権の帰属　その2

熊谷信太郎（弁護士）

他人にプログラムを作成させた場合に，プログラム著作権が誰に帰属するかについては，(1)他人に作成を指図した者にプログラム著作権が帰属する場合，(2)実際にプログラムの作成に携わった者にプログラム著作権が帰属する場合，(3)両者に著作権が帰属する場合，の３つのケースがありましたが，今回は，(2)と(3)のケースを検討します．

■(2)のケース

他人にプログラムの作成を指図した場合，指図した者が実際に創作に携わった者を，その意のままに，あたかもその手足のごとく使って著作物を完成させた場合には，指図した者にプログラム著作権が発生しますが，単に企画し，資料を提供し，あるいは注文をつけてプログラムを作成させたにすぎない場合には，指図した者ではなく，実際にプログラム作成に携わった者にプログラム著作権が発生すると解されています．これが，前述の(2)のケースの場合です．

判例には地図の作成に関して，「提供された資料や調査結果にもとづき，また指図した者の意図に沿うよう，その指図，注文をできるだけ取り入れたとしても，これを実際に創作するにあたっては，創作する者が，その感覚と技術を駆使し，自らの創意と手法により創作するものであるから，このようなものを指図した者の精神的作品とみることはできない」としたものがあります．この判例は地図の作成に関するものですから，プログラムの作成に関しては，異なる判断が必要な場合もありますが，通常はプログラムの作成を指図した者は，それだけでは著作権を取得しないといってよいでしょう．

■(3)のケース

もっとも，プログラムの作成に関する指図の内容や指図の程度によっては，指図した者が共同著作者になることはありえます．これが前述の(3)の場合です．

その場合，どの程度の指図があれば共同著作となるかは，具体的なケースごとに個別に判断せざるをえませんが，一般的にいえば，プログラム作成を指図された者が，その感覚と技術を駆使し，自らの創意と手法により主としてプログラムを創作したといえるかどうかにかかっているといえるでしょう．

つまり，そのようにいえる場合には，プログラムを実際に作成した者だけに著作権が帰属し，そうとはいえない場合，すなわち，プログラム作成を指図した者にもプログラム製作についての創作性が認められる場合には，共同著作者となり，お互いに著作権をもつことになります．

(3)のケースの例としては，友人と共同でプログラムを作成するような場合があげられます．この場合，一方が他方に指図し，注文をつけることがあっても，互いに対等の立場で共同作成作業に関与している場合には，(3)の共同著作者となります．共同著作物の著作権は全著作者の共有となり，全著作者が相談して，共同でその権利を行使することとなっています*1．したがって，著作権を譲渡したり，担保に入れたりするには全員の同意がなければなりま

Illustration : NOBUKI FUKUHARA

ワンラインNEWS
富士通は92年１月より米国で1Mビット DRAMを２割増産する．4M市場の立ち上がりが遅れているものの，依然1Mの需要が高いことから増産に踏み切った．

せん．

また，その保護期間については，共同著作者のうち最後の者が死亡した時から50年間とされています*2．

■どちらのケースに属するか

(2)と(3)のどちらのケースに属するかについて，当事者間に争いが起き，協議で話し合いがつかない場合には，裁判で決着をつけざるをえないことになりますが，(2)と(3)の区別はなかなか微妙なので裁判での立証もそう簡単ではありません．

したがって，友人などと共同でプログラムを作成する場合には，紛争になることを避けるという意味でも，事前に契約書を作成し，著作権の帰属をどうするかについて，明確に定めておくのが賢明です．

また，複数の者が共同で著作物を製作した場合でも，政府が刊行する白書や，会社の業務として著作物を製作する場合には，実際に共同製作した者ではなく，国や会社に著作権が帰属することになります*3．これはプログラムの著作権についても同様です．

＊1　著作権法第65条
＊2　著作権法第51条第2項
＊3　著作権法第15条

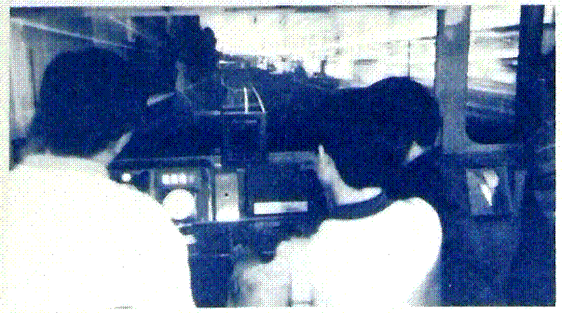
アキバ短信

交通博物館

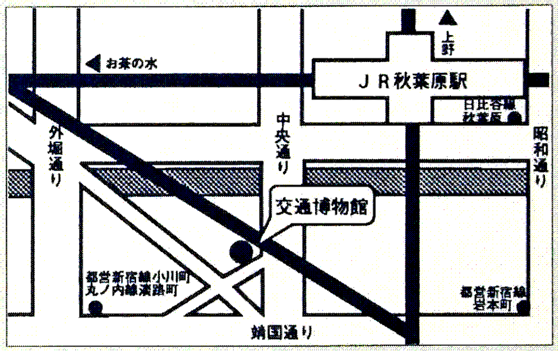
必見！　大人も喜ぶ山手線運転シミュレータ

先月号で青果市場跡の広大な空き地について触れたが，どうやらここは単なる遊休地というわけでもないようだ．去る10月に行われたF1日本グランプリの前日と前々日には，この青果市場跡地で「ミニF1」なるイベントが開かれ，なかなかのにぎわいを見せたという．いまや子供から大人まで幅広い人気を集めているF1の人気に便乗したのではあろうが，所詮1日2日のイベントである．ところが，秋葉原には「ミニF1」よりもはるかに息が長く，多くのファンを持ち続けている場所がある．万世橋のほとりにある交通博物館である．

交通博物館では，その名の通り自動車，鉄道，船舶，航空機などの交通機関全般に関するさまざまな展示が行われている．とりわけ前身が鉄道開業50年記念事業として開かれた鉄道博物館であったという経緯もあって，鉄道関係の展示は抜群に充実している．「ああ，そういえば小学校のころ，学校で連れて来られたっけ」という人も多いだろう．でも，だからといって足を運ぶのをためらう必要はない．



開館時間　9：30～17:30（入館は16:30）
休館日　毎週月曜日（祝日・休日の場合は火曜日休館）
　　　　年末年始（12月29日～1月3日）
TEL　03-3251-8481（代）



あなたが小学校を卒業して大人になるまでの間に，日本の交通機関は大幅に進歩したのだから．

博物館というからには当然古い型の蒸気機関車なども過去の遺品ともいうべき展示があるのは当然だが，ここには「東北・上越新幹線　安全体験運転シミュレータ」など最新の設備に関するもの，しかも直接自分で操作を体験できるものが多いのが特徴だ．

なかでも注目度ナンバー1は山手線運転シミュレータ(正式名称は205系電車運転シミュレータ)である．JR山手線の運転席に座って，電車の発進・停止の操作を行うのだが，本物そっくりに作られた座席の正面には実際の運転席から撮影した映像が映し出され，列車のスピードに合わせて映像が次々と変わっていく．もちろん列車の振動音も聞こえるし，ブレーキをかけるタイミングを誤ると，駅のホームの途中で停車してしまったりもする．運転時間は約3分．そのリアルさは大人でも十分楽しめるレベルである．

入場料はわずか260円(中学生以下150円)．コーヒー1杯分足らずの値段で十二分に楽しめる．1度のぞいてみてはいかがだろう．

●プログラマの仕事場 10

Programmer's Working Space

株式会社LIFEBOAT

世界中のソフトウェアを日本の市場に投入するソフトウェアマーケッター

株式会社LIFEBOATは，Lattice CコンパイラやRUNシリーズ，フルスクリーンエディタREDなど各種の言語処理系やさまざまなプログラミング用ツールの発売元として知られている．プログラマの仕事場第10回は，MS-DOSプログラム用逆アセンブラSOURCERやDOSエクステンダーDOS/4Gなど，最新の技術に注目した新製品を市場に送り出している同社をたずねて，世界各国の開発用ツールを次々と日本市場に投入する同社の秘密をさぐってみたい．

設立以来国際色豊かだが業務の中心は東京で

株式会社LIFEBOATは1981年，米国Lifeboat Associates社の日本代理店として設立されたが，実際にはLifeboat Associates社と日本のソフトウェア会社とのジョイントベンチャーのような形でスタートしたのだという．設立後，日本側で株式を取得し，すぐに米国との資本関係はなくなったため，実質的には日本の会社である．

現在は，香港にLifeboat Asia Pacificという本部オフィスを開き，その傘下に韓国LIFEBOAT社，中国とのジョイントベンチャーとして上海LIFEBOAT，米国カリフォルニア州に，LIFEBOAT SOFTWAREなど世界各国に拠点を設けており，今年7月には技術開発子会社として東京にライフボートエンジニアリング社を設立するなど，世界各地で事業を展開しているが，事実上の業務の中心となるのは東京のLIFEBOATである．

'91年11月現在で，社員数は30名，そのうち12名がエンジニアとして採用されているのだが，同社のユニークな点は開発そのものを自社内ではいっさい行っていないということだ．同社にはほかに5名のサポートスタッフがいて，製品に関する質問などを受け付けているが，移植作業などを社内で行うことはないという．

プログラマはひとりもいない!?技術者がマーケティングを担当

それでは，LIFEBOATの技術者はどんな仕事をしているのだろう．同社の製品第4部部長の刀根繁氏によれば「製品を見つけてきてEvaluate(評価)し，開発元と契約を交わして，移植情報を提供して，必要があればマシンを貸し与えて，また必要があれば，先方まで出かけていって，スペックについての打ち合わせを行って，日本での製品化にあたってマニュアルを日本語化したり，動作の確認をし，販売価格を決めて，広告を出し，PERSPECTIVEに紹介の文章を書いて，出荷する．そして出荷の後はサポートにまわる，といったことをやっています．いわゆるマーケティングをすべてやっているわけです」．

技術スタッフとして入社した，製品

ワンラインNEWS
富士通は64階調色表示のできる30インチカラープラズマディスプレイを開発した．AC型としてははじめてカラー表示に成功しているのが特徴

第４部の小松誠氏は次のように語ってくれた.「僕は最初はアルバイトで入って，サポートセンターにいました．サポートセンター時代はユーザーのプログラムをチェックしたり，時給というよりはコンパイルの回数が給料になったというくらいでしたが，入社してからは変わりましたね．思っていた仕事ではなかったです．仕事としてはもちろんコンパイルスイッチや，アセンブラにどう落ちるのか，どこにバグがあるかとか，競合はどこか，そういったことは知らなくてはいけないんですが，べつにコンパイルするわけではないのです．ユーザーからの質問には答えられなくてはいけないので，技術はもっている必要があるのですが，プログラムを組んでいるわけではありません．移植も直接はやらないのでソースも見ません.」

LIFEBOATが販売するソフトウェアには，PC／ワークステーション用のコンパイラなど言語製品，組み込みシステム開発用の各種ツール，VAX/VMSのディスク管理，メモリマネジメント，セキュリティ，PC／ワークステーション用のCASEツールや，DOSエクステンダなどの開発ツール，ライブラリなどがあるが，ほとんどすべて外国の製品であり，日本国内で利用できるよう移植や日本語化を行い，日本の市場に投入するという形をとっている．

刀根氏の説明によると，日本のパソコンへの移植は基本的には開発元で行うとのこと．製品の中にはLIFE-BOATでソースコードをひきとっているものもあるが，バグの修正やバージョンアップのたびに双方でダブルメンテナンスを行うことになっては効率が悪いので，基本的にはソースはもたないことにしているという．

問い合わせには必ず応答する欧州のソフトウェアメーカー

契約を結ぶ相手は，アメリカだけでなくアジア，ヨーロッパ各国にもおよぶ．相手がアメリカの会社の場合とヨーロッパの会社の場合では，民族性の違いからか，対応もかなり違ってくるという．自身も前日にスウェーデンへの出張から帰ってきたばかりという製品第２部部長の鈴木孝則氏はアメリカとヨーロッパのソフトウェアメーカーの違いをこんな風にたとえてくれた．「ヨーロッパの会社はたいへんまじめに仕事をやりますね（べつにアメリカがまじめにやっていないというつもりはありませんが），問い合わせた内容は必ずきっちり返ってきます．アメリカの方の会社だと，問い合わせても原因がバグらしいときなどは，返ってこないことも結構ありますよ」．

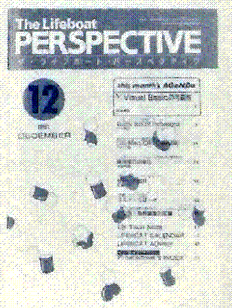
「アメリカの会社にきいたときの方が納期は早いのですが，本当にそれが守れるかどうかは神のみぞ知るというところがありますが，ヨーロッパ系だとコンパイラはいつまで，デバッガはいつまでというスケジュールを決めて，担当者を決めてということをきっちりやりますので，日本国内の商売の関係で『もっと早くほしい』とせっついてもなかなか早くはできません．でも，最初に決めた納期より遅くなることはありませんね」．

電話をとれば相手は英語最初は電話恐怖症に

仕事がら，海外のソフトウェアメーカーとの行き来やコミュニケーションの機会は多い．アルバイトの中には，

About Company

株式会社LIFEBOAT．代表取締役田先政秀．1981年に米国Lifeboat Associates社（80年代後半に消滅）の日本代理店として設立された．現在はLIFEBOAT ASIA PACIFIC LTD.を親会社とし，グループ会社として，ライフボートエンジニアリング株式会社，SHANGHAI LIFEBOAT SOFTWARE Co.Ltd.（中国 上海市），LIFEBOAT Inc.(韓国 ソウル市)，LIFE-BOAT SOFTWARE INC.(米国 カリフォルニア州）がある．パソコンユーザーには一世を風靡したCコンパイラ「Lattice C」の日本での発売元として知られている．1985年夏から「月刊The Lifeboat PER-SPECTIVE」という独自の情報誌の発行なども行っている．社員数は1991年11月現在で30名．そのうち12名が技術担当者である．



Products

「WATOCOM C/386」(シングルライセンス198,000円，コーポレートライセンス594,000円)

Intel80386/486プロセッサの持つ32ビットネイティブモードのオブジェクトコードを生成できるCコンパイラ．1パッケージにリアルモード用とプロテクトモード用の２種類のコンパイラを含む．32ビットDOSエクステンダ「DOS/4G」「386/DOS EXTENDER」をサポート．最大4GBまでのメモリを使用するアプリケーションプログラムの開発が可能である．NEC PC-9800シリーズ，PC/AT互換機（東芝J-3100シリーズ，各社AX，IBM PS/55シリーズ）の80386/486CPU搭載機種に対応．

「DOS/4G」(シングルライセンス880,000円，コーポレートライセンス2,640,000円)
VCPI, XMS, DPMIなど各種のインターフェース規格に準拠した32ビットDOSエクステンダ．32ビットネイティブのアプリケーションをデバッグできる高機能ソースレベルデバッガD32や，32ビット専用リンカGLU32が標準添付されている．

「WATCOM COBOL」(58,000円)
COBOLの標準規格であるANSI X3.23-1974, ISO 1989-1978, JIS X3002-1980に準拠したCOBOLインタプリタ．漢字をフルサポート，エラーメッセージなども日本語表示，豊富なサンプルプログラムを含み，対話型のデバッグモードを内蔵している．初心者のCOBOLプログラミング学習向き．NEC PC-9800シリーズ，東芝J-3100シリーズ，富士通FMRシリーズに対応．

「PVCS」(MS-DOS版OS/2版とも98,000円)
バージョン管理ツール．PVCSにファイルを登録すると，専用の履歴ファイルが作成される．ユーザーは自由にバージョンが設定できるため，即座に指定したバージョンのファイルを取り出せる．テキストファイルだけでなく，文章ファイルやデータベースファイルなどのバージョン管理も可能．LAN対応版もあり．

「SOURCER」(38,000円)
MS-DOS実行形式ファイルから，処理内容に対応したコメントを伴ったアセンブラソースリストを作成するコード解析ツール．8086/87/88, 80186/188, 80286/287, 80386/387, 80486, V20/V30の命令セットに対応．COMファイル，EXEファイル，デバイスドライバ，オーバーレイファイルなどのプログラムに対して，ユーザーが指定する任意のアドレス範囲への逆アセンブルが可能．各種アセンブラ処理系に特定対応したソースリスト作成ができる．NEC PC-9800シリーズ，東芝J-3100シリーズ，富士通FMRシリーズ，松下Panacom Mシリーズ，IBM PS/55シリーズ，各社AXなどで動作する．

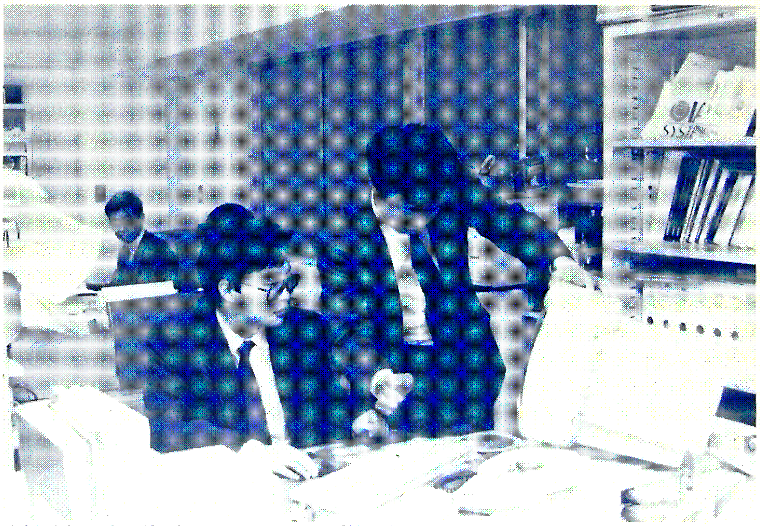
いきなり英語でまくしたてられてしばらくの間，電話恐怖症になってしまう人もいるとか．当然，かなりの語学力が要求されるのではないかとと，たずねてみると,「技術者というのはたいてい理系出身者ですから，英語なんかやりたくない，あるいは英語ができないために理系にいったというのが，よくあるタイプです．うちに来ている人もそういうタイプが多いのです．でも，入ればやらざるをえないですから」(刀根氏) との答えが返ってきた．

会社によっては，社員の語学力アップのために英会話学校へ通うための補助金を出したり，会話の先生を呼んだりするところもあるようだが，LIFEBOATの場合はもっと実戦的な方法，いきなりひとりで相手先へ海外出張に行かせるなど「セルフ・インプルーブメント」方式をとっているという．前出の小松氏も「僕は全然ペラペラではないから……」と謙遜しながらも，先ごろCOMDEX, UNIX EXPO, AXコンベンションなどのイベントを巡り，ベンダーを訪問する「ニューヨーク・ラスベガス2週間」の出張に行ってきたばかりだそうだ．

また，同社では会社の費用でコンピュータ関連雑誌を定期購読し，社員の自宅に送付している．読んでいない雑誌が目の前に山のようにあれば，こんなに情報を見逃しているのだというプレッシャーになるからだという．これも「セルフ・インプルーブメント」方式のあらわれといえるだろう．

英文のファクシミリが行き交い，電話をとれば突然英語が飛び込んでくる．子会社のライフボートエンジニアリング社には，グループ会社であるSHANGHAI LIFEBOAT社の技術者も働いている．そのひとりである唐暁丹氏は，日本で働く理由について「中国はIBM PCばかりですから」と来日してわずか3カ月というのに，しっかりした日本語で答えてくれた．ライフボートエンジニアリング社には，来年さらに大勢の技術者を中国から受け入れる予定があるそうだ．

全世界的な視野で，プログラム開発環境をとらえる同社から，今度はどんな製品が出てくるのか，期待をもって見つめたいと思う．



小松氏とライフボートエンジニアリング社の唐氏

ワンラインNEWS
日本電気は米ミップス・コンピュータ・システムズが開発した64ビットRISCチップ「R4000」を生産する．年度内に量産体制に入り，月1万個を生産する．

information・page

イベント

■1月22日(水)～1月25日(土)
第21回インターネプコン・ジャパン'92
エレクトロニクス製造技術展
第9回エレクトロテスト・ジャパン'92
エレクトロニクス設計・検査機器の専門展
第7回ADEEジャパン'92
IC/PCB設計のためのCAD/CAE関連機器展
会場：日本コンベンションセンター（幕張メッセ）　入場料：3,000円
問い合わせ先：カーナーズエクスポジションジャパン　TEL 03-3349-8501

■1月29日(水)～1月31日(土)
PAGE92
エレクトロニックプリプレス，コンピュータパブリッシングなどの関連製品と最新技術の展示会．
会場：池袋サンシャインシティ（東京）　入場料：500円
問い合わせ先：(社)日本印刷技術協会　TEL 03-3384-3411

海外イベント

■1月12日(日)～1月15日(水)
MACWORLD Expo/SanFrancisco
Macintosh関連製品の展示と，デザイン，マルチメディアなど11のコンファレンスが開催される．
会場：Moscone Centerほか（米国・カリフォルニア州サンフランシスコ）
問い合わせ先：IDGワールドエキスポ・ジャパン　TEL 03-5276-3751

セミナー

■1月8日(水)～1月10日(金)
第33回プログラミング・シンポジウム
ソフトウェア全般に関するシンポジウム，ポスターセッションなどが行われる．
会場：箱根ホテル小涌園（神奈川県）
参加費：35,000円（一般・会員），25,000円（大学院生・会員），40,000円（一般・非会員），30,000円（大学院生・非会員）（宿泊費，食費などを含む）
問い合わせ先：(社)情報処理学会　TEL 03-3505-0505

■1月21日(火)～1月23日(木)
オブジェクト指向設計とC++セミナー
オブジェクト指向技術によるC++プログラミングを体験学習する．
受講料：190,000円（3名以上，2講座受講者に割引あり）
問い合わせ先：ソフトウェアテクノロジーインターナショナル　TEL 03-5689-3536

Microsoft University

■1月13日(月)～1月14日(火)
C言語入門
QuickCを使ったC言語の入門コース．
受講料：60,000円（2日間）

■1月20日(月)～1月24日(金)
MS OS/2 Presentation Manager Programming Environment
OS/2 PMアプリケーション開発のための入門コース．
受講料：180,000円（5日間）

■1月20日(月)～1月24日(金)
Microsoft Windows Programming Environment
Windowsアプリケーション開発のための入門コース．
受講料：180,000円（5日間）

■1月27日(月)～1月31日(金)
MS OS/2 Applications Programming
OS/2アプリケーション開発のための上級コース．
受講料：180,000円（5日間）

■1月27日(月)～1月31日(金)
Microsoft LAN Manager Applications Programming
LANマネージャアプリケーション開発のための上級コース．
受講料：180,000円（5日間）

■1月21日(火)～1月24日(金)
MS OS/2 Programming Environment
OS/2アプリケーション開発のための入門コース．
受講料：150,000円（4日間）

■1月27日(月)～1月31日(金)
Microsoft Windows Applications Programming
Windowsアプリケーション開発のための上級コース．
受講料：180,000円（5日間）

■1月27日(月)～1月29日(水)
C言語応用
MS-C 6.0を使ってC言語の高度な技術を学ぶ実習中心のコース．
受講料：90,000円（3日間）

Microsoft Universityは，OS/2，Windowsアプリケーション開発者のために定期的に開催されているセミナーです．
問い合わせ先：富士ソフトウエア(株)品川教育センター　TEL 03-5461-8296

NEWS & NEW PRODUCTS

マイクロソフト，MASMとFORTRANの新バージョンを発売

マイクロソフト(株)(インフォメーションセンター：TEL 03-3363-1201)は，PC-9800シリーズに対応した「Microsoft Macro Assembler Professional Development System Version 6.0」(以下MASM6.0)と「Microsoft FORTRAN Professional Development System Version 5.1」(以下MS-FORTRAN 5.1)を発売した．

「MASM 6.0」(写真)は，プログラマの生産性を大きく向上させる新しい開発環境「プログラマーズ・ワークベンチ」を装備し，エディタ，デバッガ，NMAKEユーティリティなどのツール類を統合し，生産性を向上させる複数言語を使用したプログラミングが可能となった．他言語のプロシージャやOS/2，WindowsのAPIファンクションの呼び出しを容易にするINVOKE，PROTOディレクティブを追加，nパスアセンブラの導入によるアセンブル速度の向上，処理スピードをアップするfarCALLの最適化，80486の命令をサポートしている．また，付属の「コードビュー」は，デバッグ機能とユーザーインターフェースを改善し，複数言語の使用時でもソースコードレベルでのデバッグ，EMS，XMSメモリのサポートにより従来以上に大きなプログラムのデバッグが可能になった．価格は50,000円で，5.1ユーザーには18,540円，4.0以下のユーザーには25,750円でバージョンアップを行う．

「MS-FORTRAN 5.1」は，ソースコードを変更せずにMS-DOS用アプリケーションをWindows用アプリケーションに変換でき，Windows APIを使わずに子ウィンドウの作成，メッセージボックスの表示，クリップボードの利用が可能になる「QuickWin」機能，MASM 6.0同様，「プログラマーズ・ワークベンチ」を装備している．複数言語によるプログラミング，MS OS/2上のプログラム開発のための統合開発環境を提供している．また，付属の「コードビュー」も，XMSメモリのサポート，ユーザーインターフェースの改良などの機能強化がなされた．ANSI 77 FORTRAN，Fortran90の拡張やDEC，VAX，IBM SAAなどの言語使用を採用している．価格は98,000円で，4.1および4.0ユーザーには25,750円，3.31以下のユーザーには36,050円でバージョンアップを行う．

また，両製品のFM Rシリーズ，FM TOWNS対応版は92年3月上旬に富士通㈱から提供される予定．

CSK，LAN対応パソコン用アプリケーション開発ツールを発売

㈱CSK (TEL 03-5563-9300)は，マルチユーザー，マルチベンダー環境下でビジネスアプリケーションの構築を支援する「CS-CONCEPT LAN 3.0ver.」を発売した．従来のスタンドアロン版に，LANに対応する機能として，排他制御機能，異機種接続におけるプリンタの共有などをサポートしている．アプリケーションの開発は，システムメニューを利用して必要な処理を選択し，各種パラメータを設定するだけで簡単に行え，入力画面や出力帳票はワープロ感覚で定義し，設計することができる．また，定型，非定型の豊富なユーティリティのほか，C＋BASICライクな開発言語「LUI」も用意されている．当初の対応機種はPC-9800シリーズ，対応LANはNetWare，LANマネージャ，Net/One，NFSなど．価格は，アプリケーション開発・実行環境版が198,000円，アプリケーション実行環境版は86,000円．また，従来のスタンドアロン版からのバージョンアップも行う予定．

ワンラインNEWS
川崎製鉄はHDDと光磁気ディスク装置を組み合せたワークステーション向け記憶装置を開発，販売を開始した．HDDの容量が45MBと325MBの製品がある．

NEWS & NEW PRODUCTS

富士通，ディスプレイ一体型モデルなど「FM TOWNSIIシリーズ」を発売

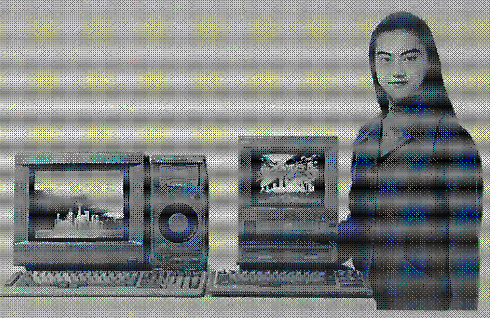
富士通㈱（FM TOWNSインフォメーションサービス：TEL 03-5690-0032）は，ハイパーメディアパソコン「FM TOWNS」の新シリーズとして，2機種6タイプの「FM TOWNS II シリーズ」を発売した．高精彩10インチディスプレイ一体型の「FM TOWNS II モデルUX」（写真右）は，従来の約半分の設置スペース，1フロッピーディスクドライブモデルで263,000円という低価格を実現した．「同モデルCX」（写真左）は，従来機種の約1.4倍の高速処理を実現し，100MBハードディスク搭載モデルをラインナップに加えるなど，高速・大容量化をはかった．付属の「Townsシステムソフトウェア V2.1」は，ウィンドウ表示の採用により操作性を向上，高機能住所録データベース機能を備えたハガキ作成ソフトやCD辞書検索ソフトなどの実用ソフト群「Towns Staff」，アプリケーション実行時に予定表などを呼び出せる「サイドワーク機能」などを標準装備．また，NetWare386 v3.1JなどのLANをサポートしている．

■FM TOWNS II モデルUX10/UX20

プロセッサ：80386SX（16MHz）／メモリ：2MB（最大10MBまで拡張可能）／外部記憶装置：3.5インチFDD×1（UX10），3.5インチFDD×2（UX20），CD-ROMドライブ×1（全モデル）／価格：263,000円（UX10），288,000円（UX20）

■FM TOWNS II モデルCX10/CX20/CX40/CX100

プロセッサ：80386（16MHz）／メモリ：2MB（最大16MBまで拡張可能）／外部記憶装置：3.5インチFDD×1（CX10），3.5インチFDD×2（CX20），3.5インチFDD×2・40MB HDD×1（CX40），3.5インチFDD×2・100MB HDD×1（CX100），CD-ROMドライブ×1（全モデル）／価格：30,8000円（CX10），333,000円（CX20），433,000円（CX40），533,000円（CX100）



ロンローインターナショナルネットワークス，DOS/V対応超小型マシンを発売

㈱ロンローインターナショナルネットワークス（TEL 03-3493-5850）は，IBM PC/ATのすべての機能を備えた世界最小のデスクトップ型パーソナルコンピュータ「Tinyシリーズ（620/650）」を発売した．小型，軽量，低価格であることを目的に設計され，フロッピーディスクドライブ程度の大きさ，218mm×78mm×134mm，重さ1.7Kgを実現している．グラフィックスはVGAを採用し，DOS/V対応の650モデルはマウスを標準装備している．

■Tiny650 SF/DF/HF

プロセッサ：80286（12MHz）／メモリ：1MB（最大4MBまで拡張可能）／外部記憶装置：3.5インチFDD×1（SF），3.5インチFDD×2（DF），3.5インチFDD×1・40MB HDD×1（HF）／価格：108,000円（SF），128,000円（DF），168,000円（HF）

■Tiny620 SF/DF/HF

プロセッサ：V30（10MHz）／メモリ：1MB／外部記憶装置：3.5インチFDD×1（SF），3.5インチFDD×2（DF），3.5インチFDD×1・40MB HDD×1（HF）／価格：78,000円（SF），98,000円（DF），138,000円（HF）

日本電気，98用ネットワークサーバコンピュータを発売

日本電気（株），日本電気ホームエレクトロニクス（株）（NECパソコンインフォメーションセンタ

NEWS & NEW PRODUCTS

ー：TEL 03-3452-8000）は，サーバコンピュータの新シリーズ「98サーバシリーズ」の製品化を決め，第一弾として，最下位モデル「NECパーソナルコンピュータサーバSV-H98モデル30」を発売した．同シリーズは，アプリケーションソフトの開発を容易にするため，PC-9800シリーズのハードウェア，ソフトウェアの設計思想を使用しながら高速性，システムの拡張性などを追求していく．日本語MS OS/2 LANマネージャ，NetWare386などのネットワークOSに対応，床置き型の採用により，32ビットバスアーキテクチャ「NESA」対応の拡張スロットを8スロット内蔵，固定ディスクを最大3台まで内蔵することができる．

■SV-H98モデル31/32/33

プロセッサ：80486（33MHz，ハイレゾモード）／メモリ：7.5MB（最大31.5MBまで拡張可能）／外部記憶装置：300MB HDD×1（モデル31），600MB HDD×1（モデル32），600MB HDD×2（モデル33）・3.5インチFDD×2（全モデル）／価格：2,230,000円（モデル31），2,680,000円（モデル32），3,530,000円（モデル33）

アイシーエム，内蔵型HDD，光磁気ディスクなどの新シリーズを発売

㈱アイシーエム（大阪営業所：TEL 06-648-4702）は，同社の内蔵型ハードディスク「INTERシリーズ」，ノートPC用ハードディスク「Packシリーズ」を全面的にモデルチェンジし，ハードディスク4シリーズ12モデル，光磁気ディスク1モデルの計13モデルを発売した．PC-9800シリーズ用内蔵ハードディスク「INTERシリーズ」には，300MB，500MBの大容量モデル「Wシリーズ」（300MB：398,000円・500MB：548,000円），1MBキャッシュメモリを搭載した「WSシリーズ」（100MB：128,000円・130MB：178,000円），NEC純正のドライブを使用した「WNシリーズ」（100MB：128,000円）を，ノートパソコン内蔵型の「Packシリーズ」には，80MBのパラメータをもたないPC-9801NS/Eなどでも使用できる「Pack E-80」（80MB：128,000円）などを追加した．3.5インチMOディスクドライブユニット「MO-3120」は，約120MBの記憶容量をもち，ディスク回転数3000回転／分，平均シーク時間40ms以下と，ハードディスク並みのスペックを実現している．本体価格は198,000円，ケーブル，ターミネータなどが付属した接続キット（PC-9800シリーズ用・Macintoshシリーズ用）は30,000円．

メルコ，シリコンディスクSDRシリーズを低価格で発売

㈱メルコ（メルコ・インフォメーションセンター：TEL 052-251-8365）は，高密度実装技術により，1スロット100MBの大容量を実現し，40MBで1か月，100MBで2週間のバッテリーバックアップが可能な「シリコンディスクSDRシリーズ」を発売した．シリコンディスクは外部拡張スロットに装着するもので，物理的稼働部がないため，耐震性にすぐれ，塵や埃にも強くFAニーズにも対応することができる．4MB～100MBで，価格は69,800円～550,000円．また，エプソンPC-386用の内部増設RAMボード「ERB-4000」（4MB：44,800円），増設RAMモジュール「XRBシリーズ」（1MB～4MB：11,000円～41,000円）も発売した．

コンテック，16ビットバス対応のI/O拡張ユニットほかを発売

（株）コンテック（TEL 0120-01-9804）は，AXシリーズ，J-3100シリーズ，IBM PC/AT用の16ビットバス対応I/O拡張ユニット「FA-PAC（PC）VH-5」を発売した．本体のATバス仕様拡張スロットを5スロット増設することができる．AXハーフサイズのコンパクトな筐体，マザーバスボードに4

ワンラインNEWS
アップルの発売した，同社初のノート型パソコン「Macintosh PowerBook」シリーズの最下位機種である「同100」はソニーがOEM供給を行っている．

NEWS & NEW PRODUCTS

層基板を採用するなど，耐ノイズ性についても考慮した設計となっている．価格は126,000円，フルサイズボード用の「FA-PAC（PC）H-5」128,000円もある．

セガ・エンタープライゼス，DOS/V用ソフト2種を発売

㈱セガ・エンタープライゼス（TEL 03-3743-7432）は，同社の「テラドライブ」およびDOS/Vマシン用ソフトとして，日本語レイアウトソフト「Page Pro-S」と通信ソフト「Auto Net-S」を発売した．「Page Pro-S」は，簡易デスクトップパブリッシングタイプのワープロソフトで，マウスによる自然な操作で，文字サイズの変更など，文字・図形の編集が行える．日本語入力フロントエンドプロセッサ「EGBridge」を標準装備している．価格は9,800円．「Auto Net-S」は，初心者から上級者まで使える通信ソフトで，通信速度は300～38400ボー，XMODEM，YMODEM，TRANSITなどのプロトコルをサポートし，複雑なオートパイロットも行うことができる．価格は14,800円．

インターコム，DOS/V対応のファイル転送ユーティリティ「LAPLINK-J」を発売

㈱インターコムは，DOS/Vに対応した超高速ファイル転送ユーティリティ「LAPLINK-J」を発売した．すでにPC-9800シリーズ，J-3100シリーズなどに対応していたが，新たにDOS/Vにも対応したことで，DOS/Vパソコンとも自由にファイル転送が行え，効率的なデータの共有・交換が可能となった．パッケージにはPC-9800シリーズ，J-3100シリーズ，PC-286/386シリーズ，DOS/Vパソコンに対応したプログラムと4ヘッドケーブルが含まれており，1パッケージでさまざまなパソコン同士の組み合わせでのファイル転送を行うことができる．また，販売は(株)インターソフト(TEL 03-3293-3338)が行う．価格は28,000円で，LAPLINK-Jユーザーには無償でDOS/V用プログラムを提供する．

キヤノン，アスキーの「ウィンズ」「インフォミックス」をNeXTに採用

キヤノン㈱（SH推進室：TEL 03-3344-8421）は，(株)アスキー（ビジネスソフトウェア事業部：TEL 044-952-8211）の表計算ソフトウェア「ウィンズ」とリレーショナルデータベース（RDB）「インフォミックス」をNeXT用ソフトウェアとして商品化，販売を行うことで合意し，アスキーが日本語化，キヤノンが商品化を行う．「ウィンズ」は，高度なグラフィックス機能と強力なアプリケーション開発言語を搭載している．価格は168,000円，1992年3月に発売の予定．「インフォミックス」は，日本市場におけるUNIXベースのRDBでは第1位のシェアをもつ分散型RDB．価格は102,000円～4,250,000円，同年5月に発売の予定．これを機にキヤノンでは法人マーケットを中心としたOA分野にNeXTを本格的に投入し，市場拡大をはかる．

カノープス電子，画像通信を可能にする画像圧縮ツールを発売

カノープス電子(株)（カノープステクニカルサポート：TEL 078-412-7166）は，パソコン通信で自然画の通信を可能にする画像圧縮ツール「JANUS-NET」を発売した．フレームメモリ上の自然画を国際標準規格JPEG方式で効率よく圧縮，伸長を行い，圧縮時には9通りの圧縮率を指定することができる．圧縮画像をPC-9800シリーズのVRAMに表示するコマンドが用意されており，フレームメモリがなくてもダウンロードした画像を表示することができる．また，Macintosh上で作成したJPEG準拠圧縮画像ファイルを利用することもできる．価格は8,000円．

ワンラインNEWS
シャープはOADG仕様の32ビットノート型パソコン「PC-6700」シリーズを1月より販売する．同製品はHDDとFDDの両方を内蔵したノート型では世界最小・最軽量．

Programmer's Program

True BASIC用ライブラリ ビジネス用グラフィックスツールキット/3次元グラフィックスツールキット/科学技術用グラフィックスツールキット/数学者用ツールキット

「本物のBASIC」を謳って発売されたTrue BASICはANSI規格に準拠し，シンプルな命令体系によるわかりやすいプログラミングを特徴にしている．今回紹介する4種類のツールキットはTrue BASICをそれぞれの分野で強力に支援するためのライブラリである．

各ツールキットの特徴

【ビジネス用グラフィックスツールキット】

ビジネス用グラフィックスツールキットは，17種類のビジネスチャートを高速描画できる，TrueBASIC専用のツールキットである．棒グラフ・線グラフなど単一のグラフはもちろん，多種類のグラフを1枚の複合グラフとして表示するなど，多彩な表現でビジネスチャートが作成できる．

【3次元グラフィックスツールキット】

3次元グラフィックスツールキットは，独自の3次元ウィンドウを設定するだけで，簡単にしかも正確なグラフや図形を描画することができる．複雑な手続きや設定を必要としないため，はじめて3次元グラフィックスにふれる人でもとまどうことはない．特殊な3次元アニメーションや複雑な立体グラフなど，それぞれの目的に合わせて応用することができる．

【科学技術用グラフィックスツールキット】

科学技術用グラフィックスツールキットは，科学の分野で必要とされているさまざまなグラフと座標形式を取り揃えている．対数グラフはもちろん，ベクトルグラフや極座標グラフなど，まさに科学データを120％生かすためのツールキットである．これまで，膨大な時間のかかっていた机上シミュレーションなども，簡単な操作で実現できるようになる．

【数学者用ツールキット】

数学者用ツールキットは，10万桁の演算や複素行列など，通常では処理できない複雑な演算を行うことができる．高い精度でデータを求めたいときや，大きな桁数の数値を扱う場合などに，抜群の威力を発揮する．

教育を意識したガイドブック

各ツールキットのガイドブックは教育をかなり意識しているのがわかる．たとえば3次元グラフィックスツールキットには，サンプルの説明と各サブルーチンの説明の間に入門者にも理解できるよう3次元グラフィックスの理論や投影法などが12ページにわたって解説されている．各サブルーチンもレベルごとに分けて解説され，順を追って読んでいけば理解できるように工夫されている．

またガイドブックの付録にはサブルーチン一覧やエラーメッセージ一覧が掲載され，プログラムを書くときや，エラーが出たときにもひきやすいよう配慮されている．

インストール

インストールは，各ツールキット共通で，付属のガイドブックにしたがって操作するだけである．ハードディスクや2ドライブフロッピーディスクの場合だけでなく，ノート型パソコンのようにフロッピーディスク1台とRAMドライブの場合のインストールについても詳しく述べられている．

たとえば，ハードディスクの場合はTrue BASICをインストールしたディレクトリに各ツールキットのディレクトリを作成し，ツールキットのディスクにある専用のインストーラを起動する．あとはTrue BASICを起動し，True BASICにツールキットのライブラリをインストールしたディレクトリを指定すれば使用可能になる．

シンプルだが強力なサブルーチン

インストールが終了したら，まずサンプルプログラムを走らせてみる．各ツールキットには豊富なサンプルソースプログラムが付属しているので，具体的にどんなことができるかを知ることができる．サンプルプログラムは，そのデモンストレーションの内容とは裏腹にシンプルそのもので，謳い文句の「シンプルだから使える」を思い出

ワンラインNEWS
日本電気は16MビットDRAMのサンプル出荷を開始した．生産規模は月産5万個で，ワークステーションなどの需要を見込んでいる．

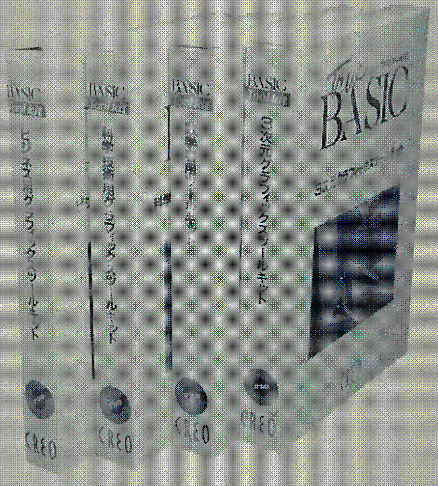
す．またソースファイルにはコメントも多く書かれているので，初心者でも理解するのに時間はかからないだろう．

True BASICはIBM PC, PS/2, その互換機，アップルMacintosh 512K, Plus, SE, IIシリーズ，Atari ST 520, 1040, MegaST, Commodore Amiga 500, 1000, 2000などのパソコンに対応し，ほかにもSilicon Graphics, Sun Microsystems, Inc., Data General Corp. 各社のUNIXワークステーションに対応している．国内では，日本電気のPC-9800シリーズ対応の「日本語True BASIC V3.0J」を㈱クレオが販売している．

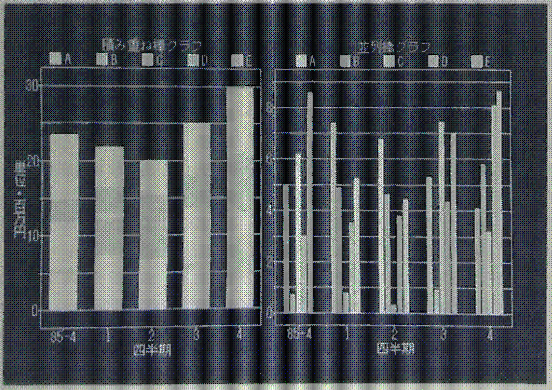
今後クレオでは「文字列操作ツールキット」，「並べ替え・検索ツールキット」，「統計用グラフィックスツールキット」の追加発売を予定している．

■ビジネス用グラフィックスツールキット
3次元グラフィックスツールキット
科学技術用グラフィックスツールキット
数学者用ツールキット

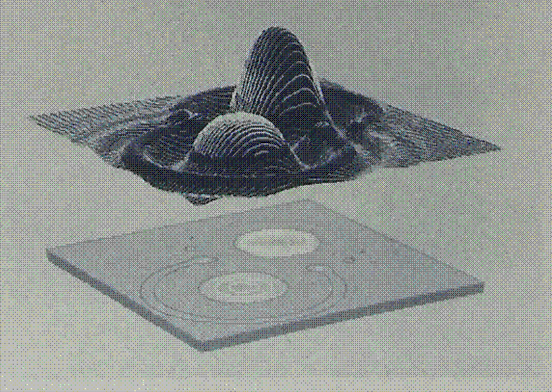
販売：（株）クレオ（TEL 03-3456-4650）／価格：18,000円（ビジネス用グラフィックスツールキット）・18,000円（3次元グラフィックスツールキット）・25,000円（科学技術用グラフィックスツールキット）・25,000円（数学者用ツールキット）（消費税含まず）／適応機種：日本語True BASICが動作する全機種／動作環境：MS-DOS Ver.2.11以降，日本語True BASIC／必要メモリ：640Kバイト／供給メディア：5インチ2HDまたは3.5インチ2HD



ビジネス用グラフィックスツールキットのグラフ



3次元グラフィックスツールキットの3次元グラフィックス



オンラインソフト採集帳 13
テキストページャー MIEL

フリーソフトウェアのドキュメントや，ログファイルなどテキストファイルの内容を見たいとき，ページャーはとても便利なツールである．テキストの編集はできないが，「見る」ための機能は一通り揃っているし，エディタにくらべすばやく立ち上がる．「MIEL」はフリーソフトウェアのファイル管理ツールである「FD」に付属していた機能を独立し，さらに多機能化したページャーである．

環境の設定

環境設定ファイルMIEL．CFGでファイル選択モードやViewモードの各キーの割り当て，画面や各ファイルの属性ごとの色設定ができる．ほかにも外部プログラムを利用して実現する機能がある．おもな設定として

EDITOR
　テキストエディタの設定
RXARC
　アーカイブユーティリティの設定，
MIELPT
　子プロセスでのプロンプト
RXFIND
　あらかじめ設定する検索文字列
RXCLIP
　表示画面をクリップするファイル名
RXTREE
　ファイル選択モードで起動するツリー表示プログラム
RXEXT
　拡張子を省略したときのディフォルト拡張子
TMP
　作業用ドライブ

などがある．これらはMIEL．CFGまたは環境変数に設定することができる．

またMIEL．CFGを

```
set_useEMS = 1
```

とすれば，ファイル読み込みバッファが増大し，またヒストリーのバッファとしてEMSのメモリを使用する．さらに検索文字列入力の最大文字数が32文字となり，ファイル選択時の［W］によるファイルマスク入力時にもヒストリーが使用できるようになる．それぞれヒストリー数としては最大64項目まで使用できるようになる．

起動する

MIELでは複数のテキストファイルの内容を同時に見ることはできない．しかし選択対象として複数のテキストファイルを指定し，ファイル選択モードで任意のテキストファイルをカーソルで選んで内容を見ることができる．たとえばMS-DOSのコマンドラインから

```
A> MIEL AAA  BBB  CCC
```

と入力すれば，それぞれのテキストファイルがファイル選択モードの対象となる．また，最初のファイルの指定にはワイルドカードの指定ができるので，たとえばMS-DOSのコマンドラインから

```
A> MIEL B:¥TOOL¥*.DOC
```

と入力し，BドライブのTOOLディレクトリにあるDOCを拡張子にもったすべてのファイルを対象とすることができる．

複数のテキストファイルを選択の対象とした場合は，起動後ファイル選択モードとなる．このモードでは対象とされたファイルの一覧のほかに，カーソル位置のファイルの先頭の内容が画面に表示されるので目的のファイルを検索する参考にすることができる．

目的のファイルにカーソルを移動し［RETURN］キーを押せば，Viewモードとなり，ファイルの内容を見ることができる．複数のファイルを指定している場合はViewモードで［ESC］または［RETURN］(デフォルト時)を押せば，再びファイル選択モードとなる．

LHAの圧縮ファイル内のドキュメントが読める

MIELの特徴の１つに圧縮ファイル内の任意のファイルを閲覧できることがあげられる．LHAで圧縮されたファイルを見るには起動時-Lオプションを指定し，圧縮ファイル中の見たいファイルの前に＋を付加する．たとえばMS-DOSのコマンドラインから

```
A> MIEL -L MIEL237.LZH +
MIEL.DOC
```

とすれば，MIEL237．LZHに圧縮されているMIEL．DOCの内容を見ることができる．さらに-NAオプションの指定でLHAで作成された自己復元型の圧縮ファイルに対しても同様のことが可能になる．MS-DOSコマンドラ

ワンラインNEWS
日本ユニシスはオフコン「シリーズ8」を一新し，接続性，データ蓄積機能などを強化した「シリーズ8・Rファミリー」11モデルを発売した．

インから

A> MIEL -L -NA LHA213.
EXE +LHA.DOC

とすれば，LHA213.EXEに圧縮されているLHA.DOCの内容を見ることができる．これは非常に便利な機能である．

あらかじめ編集しないことがわかっているテキストファイルならページャーで見るほうが何かと便利であろう．いうまでもなくTYPEコマンドとMOREを組み合わせて使うよりも便利であるし，エディタで読み込んだ場合のようにキーの押し違いで元のファイルを改変してしまう心配もない．さらに圧縮されたファイルから任意のテキストファイルを見ることができるなど，ほかのソフトウェアには見られない機能が用意されている．

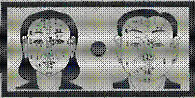
■MIEL Ver.2.37

名称：MIEL／分類：テキストページャー／登録名：MIEL237.LZH／作者：出射厚（A.Idei,NIFTY-Serve:SDI00544），清水洋平（Paci,NIFTY-Serve:RGD00165/pcs:pcs35438）／動作環境：PC-9801シリーズ MS-DOS（別にFMRシリーズ用あり）／登録月日：1991年9月18日／復元方法：LHA E MIEL237／入手場所：ASCII-NET PCS（pool msdos #4383）

ファイル構成

Listing of archive : MIEL237.LZH

Name	Original	Packed	Ratio	Date	Time	Attr	Type	CRC
MIEL.COM	18549	15041	81.1%	91-09-16	02:37:00	a--w	-lh5-	DF66
MIEL.CFG	6858	3136	45.7%	91-09-16	02:37:00	a--w	-lh5-	7C3F
MIEL.DOC	37735	12664	33.6%	91-09-16	02:37:00	a--w	-lh5-	D12E
MIELCUST.COM	3154	1958	62.1%	91-09-16	02:37:00	a--w	-lh5-	BA7C
MIELCUST.DOC	2056	966	47.0%	91-09-16	02:37:00	a--w	-lh5-	1561
MIEL237.DOC	1288	837	65.0%	91-09-16	02:37:00	a--w	-lh5-	1490
6 files	69640	34602	49.7%	91-09-18	01:39:26			

HEALTHY?  UNHEALTHY?

⑬ 電磁波の謎①

CRTディスプレイから出る人体に有害な電磁波などをカットするというフィルターや眼鏡，エプロンなどが市販されている．もともとそうした問題が発生したのは，OAブームによって，パソコンやワープロがオフィスに入るようになってからだ．つまり，80年代後半のころからだろう．

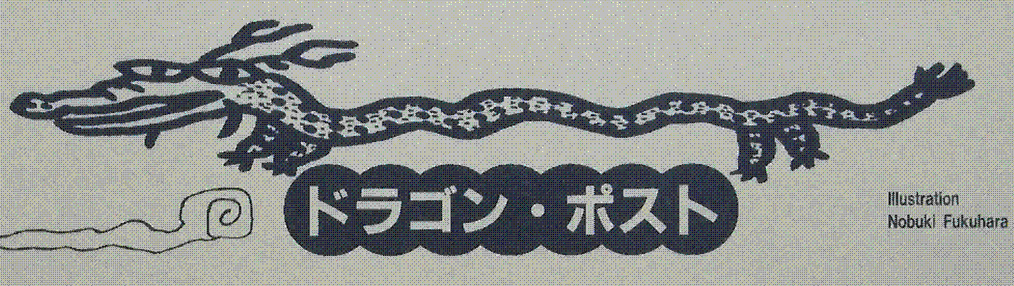
キーボード入力による腱鞘炎などは，身体の一部を酷使という理由があるからかなりわかりやすいのだが，電磁波となるとどうも理解しにくい．労働省などの公式見解としては，1時間当たり15分程の休憩をとればVDT（ビデオディスプレイチューブ）作業には問題ないということにおちついたはずだ．

それでも，「大丈夫だろうか」という人は絶えないらしく，防護フィルターを使っている人は少なくないし，先日は新聞に防護エプロンの記事が掲載されたりもした．防護フィルターなどは，大手の化学・フィルムメーカーなどか発売されているが，パッケージに入っている説明書などには，ディスプレイから何が放射されていてフィルターはどのような電磁波をカットするのかといった説明はない．こうなると何が有害なのだろうかという謎が残る．

テレビが普及しだしたとき，やはり同じような問題が起こってる．「身体に害があるから近づいて見てはいけない」「見る時間を制限する」といった具合である．これだけテレビが普及した現在でも，テレビによって身体に障害がでて問題化したというようなことはない．だからといって，CRTディスプレイが大丈夫だということにはならいが，不安に対して防護機器を出すならはっきりした説明があってもいいのではないだろうか．

CRTディスプレイは有害なのか? パソコンのヘビーユーザーに聞いてみても，はっきりした答は返ってこない．今月から数回にわたってこの問題を追いかけてみようと思う．

ワンラインNEWS
日本データ・ゼネラルはRISCチップを採用したワークステーション2機種を発売．シリーズの最下位機種で，従来機にくらべ価格性能比を最大2.5倍に引き上げている．



ドラゴン・ポスト

Illustration
Nobuki Fukuhara

この欄は，読者からのお便りを中心に構成しています．本誌に対するご意見・ご感想などをお待ちしております．鬼より恐い読者からの声ということで，「ドラゴン・ポスト」です．本誌についているアンケートはがきにご意見の欄を設けてありますから，そこに書いてください．ペンネームでも歓迎です．ただし，はがきには本名と住所をしっかり書いてくださいね．掲載誌をお届けします．

★プロでも気になる……

最近のホビープログラマの質はかなり向上してきていますね．プロのプログラマも目を見張るものがあります．そんなホビープログラマの開発環境はどんなものなのか，プロ側でも気になりますね．ちなみに私の環境は286（12MHz），13MB，OS/2，MS-C6.0です．速いマシンがほしい欲求は，常に持っていますが，作成したプログラムの質は変わらないので，当分このままでしょう．（斉藤明好）

――たしかに，アマチュアが作るプログラムの質は上がってきているようです．しかし，だからといって特別すごい開発環境をもっているわけではないでしょう．やはり，努力と忍耐，そして工夫でしょうね．

★Cだ，Cだというけれど……

プログラマではないのですが，内容を見て興味をひかれ，購入するようになりました．プログラミングはBasic位でしか作ったことがありません．CだCだと世の中C言語が全盛のようですが，Basicの方がとっつきやすい気がします．この本はプログラマしか読まないのだろうか．（毛利章二）

――Cはそっけない言語ですから，素人にはとっつきにくいかもしれません．でも，そこがシンプルでいいという人も多いんですよ．C以外の言語についても，徐々にとりあげていきたいとは思ってます．

★のぞいてみたいプロのHD

パソコンを始めてようやく1年がすぎようとしています．ソフトもいろいろとふえて，40MB HDもどこに何が入っているか，またPATHはどこまで通っているかなど，ごちゃごちゃしてきたので，一度，大そうじをしたいのですが，プロや名人の方々は，どのようなHDの利用をしているのでしょうか．一度，特集してほしいです．（山本光）

――プロが使っているHDというと，300MBとか600MBとかいうのも別にめずらしくありません．大容量HDのディレクトリを構築する際には，UNIXのディレクトリ構造に似せていることが多いようです．

★Windowsプログラミングに挑戦中!

最近，Windowsのプログラムなるものをはじめました．しかし，今までのプログラムならシステムコールという方法で，OSの資源を利用していたのに，これからはメッセージとか，しかもWindowsからよばれるのを待たなければならないなんて，私には，まったくわけのわからないことが出てきます．時代の流れの早さに頭をかかえるばかりです．（小林正登）

――メッセージパッシングはWindowsプログラミングを理解する要になる概念ですからがんばって理解してください．なに，すぐ慣れますよ．

★暴走につぐ暴走で……

先日，自作のプログラム（マシン語－死語？）で，グラフィック画面をドット単位で修正する必要にせまられ，DOS付属のMOUSE.SYSを使用していたところ……汚い！　これは我慢できない．しようがないので，某社のテクニカルデータブックをひっぱり出して，見よう見まねでプログラミングをしてはみたものの，割り込み処理なぞ，VSYNC以外やったことのない私のこと，暴走につぐ暴走で，まともなプログラムが組めずにいます．そこでお願いなのですが，今度，貴誌の企画で，ぜひこの手の割り込みに関してのプログラミングの特集を組んでほしいのです．（橋本秀俊）

――割り込み関係のプログラミングには，やっかいなものが多いんですよね．「98活用アセンブラ技法」などが参考になるんじゃないでしょうか？

★もっとハードがらみの記事を!

まだBasicやCなどのプログラム言語でしか，プログラムができないので，貴誌の内容を理解するには不十分ですが，貴誌はハードについてのBIOSの解説とかが少ないように思います．特定機種のプログラム作成には不可欠だと思うのですが．（プログラマーズページ一年生）

――ハードに密着したシステムプログラムとハードを意識せずにできるアプリケーションプログラム（ほんとかな？）の両方をバランスよく取り上げたいとは思っているのですが．

★バージョンアップしないで!!

PC-386Sにした。EPSONのDOSもかった。インストールするひまがないので，HDにはまだNECのDOSが入っている。……なのに!　もうNECは5.0にするという!!　たのむからVer Upは来年にしてくれ!　まだWINDOWSもかってないのに……!! (SHIFT)

――SHIFTさんのような人のことを「バージョンアップ貧乏」と呼ぶんですよ。なにを隠そうこの私も……;ー)

★OS/2で脳ミソが一回転!?

もう一年も前になるのですが，IBMのOS/2で医療画像システムというのを開発する機会がありました。関数を呼び出すタイミングが，メッセージという考え方ですので，今までの引数渡しによるコールになれていた脳ミソが，一回転した思い出があります。「WIN V3」にしても，メッセージの概念を理解しないと，プログラムの製造は出来ませんので，今までのDOSベースと同じイメージを持たれると苦労するのではないでしょうか。(加藤敏郎)

――そうです，皆さん苦労なさってます。古いスタイルのプログラミングが得意であればあるほど，新しいスタイルに移行できないようですね。

★ただいまナイコン状態

現在，8bit機を所有していますが，TVタイマーぐらいにしか使っておらず，実質的ナイコン状態です。いずれ新機種を買おうと思うので，その時までにいろいろな知識を身につけようと，貴誌を創刊号から読んでいます。「習うより慣れろ」とはいうものの，本体を持っていない人にもわかるように，もう少しやさしく書かれていれば……。(Ship)

――実際にプログラムを動かしながら学べないというのは，ちょっとつらいですね。できるだけわかりやすい記事をと心がけてはいるのですが，ページ数にも限りがありますし，むずかしいところです。

★キーボードの調子が……

最近，キーボードの調子が良くありません。どのような症状かというと，キーボードを押しても画面に文字が出力されないんです。それも一度に5，6個のキーがなることもあります。使っていれば，また正常な状態にもどるんですが，これは故障なんでしょうか。一度，キーボードを洗っても効果がありませんでした。故障といえば愛機98EXの3.5インチドライブは2度も故障しています。二度とも同じ症状でディスクを入れても，ディスクが入っていない状態のままなんです。私の知っている友人二人も一度故障しています(もちろん同機種)。5インチを増設して，そちらをメインで使用しているため，今のところ正常に動作しています。みなさんの中にも同じような故障の経験をなさった方はいるのでしょうか。(北Pei)

――パソコンも機械ですから，故障もあります。キーボードやフロッピーディスクドライブなどはもっとも疲弊の激しい部分ですから，一度メーカーに修理に出してみてはいかがでしょう。

★「遊」工房がたより？

Cの勉強をはじめました。main ()，#include，#define……???　本を読むだけではと，QuickCを起動する。メモリモデル？　リリース???　何をやってもようわからん。私には「遊」工房が唯一のたよりです。なぜ？　それは，ソースのほとんどの行に「/*…………*/」のコメントをつけてくださっているからです。(武田徹也)

――プログラムを覚えるには人の書いたソースを読むのがいちばんとよくいいますからね。弊社発売の「C文解釈」なんてのもどうでしょう？

★そんなお金があるわけない!

ああ!　NECのDOS 5.0はメモリが1.5Mもいるじゃないか!　ああ!!　ハードディスクもいるじゃないか!!　高校生にそんなお金があるわけない!　でもやっぱりほしいですね。(Programmer's Pageを読んでる人の平均年齢は高いのではないでしょうか？)(中村倫俊・絵も)

――高校生でパソコンをもてるだけでも幸せじゃありませんか。私が高校生の頃なんて……うぅ;ー)

★買い換えを計画中!

現在，お金もないのに買い換え(パソコンの)を計画中。本命はあいかわらずMac。でも，IBMとAppleが手を組んだというし，今後Macはどうなるのだろうか。あと，UNIXの動きも気になる。でもUNIXマシンは高価だし，一般ユーザーに簡単に探せるものでもない。どこか，一般ユーザーにUNIXマシンを使わせてくれるところはないかなあ(電話回線等＋パソコンを端末にして)。(学校にUnixがあるけどIDがない狂夢)

――ワークステーションは安くなったとはいえ，まだまだ高価ですね。いっそMacを買ってAIXというのはどうでしょう。

(敬称略)

創刊1周年記念 読者プレゼント

1 PC-9801NS/E

応募は本誌に添付のアンケートはがきをご利用ください(1992年1月17日消印有効)．なお，応募者多数の場合は，抽選になります．当選者の発表は発送をもって代えさせていただきます．

＊1：本体，キーボード，マウスをモニターとして提供します．ディスプレイ装置は含まれません．当選された方には，一定期間の使用の後，製品に関するレポートを提出していただきます．

＊2：ビジネス用グラフィックスツールキット，科学技術用グラフィックスツールキット，数学者用ツールキット，3次元グラフィックスツールキット

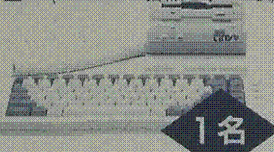
＊3：MS-C Ver.5.1以上，QuickC Ver.2.0以上に対応

＊4：3.5インチ，5インチ各5名．メディアのサイズをお書きください．

1名

■プロセッサ：80386SX（16MHz）／メモリ：640KB／外部記憶装置：3.5インチFDD×1，1.25MB RAMドライブ×1／ディスプレイ：8階調CFLサイドライト付白黒液晶ディスプレイ

4 Tiny650 ＊1
(80286・VGA搭載 PC/AT互換)



1名

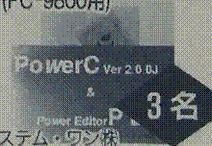
提供：(株)コンローインターナショナルネットワークス

5 日本語TrueBASIC V3.0J ＋ツールキット4本 ＊2
(PC-9800用)

1名

提供：(株)クレオ

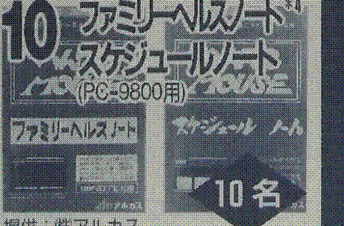
6 Power C Ver.2.0.0J&PED ＊3
(PC-9800用)



3名

提供：システム・ワン(株)

10 ファミリーヘルスノート スケジュールノート ＊4
(PC-9800用)



10名

提供：(株)アルカス

11 高機能エディタ Brief

1名

提供：日本システム(株)

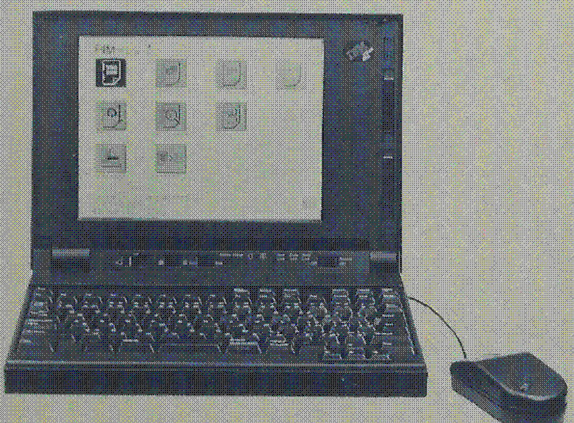
12 ビデオ「オブジェクト指向プログラミングの世界」
(VHS)

2名

提供：ボーランド(株)

2

PS/55note モデルN23sx

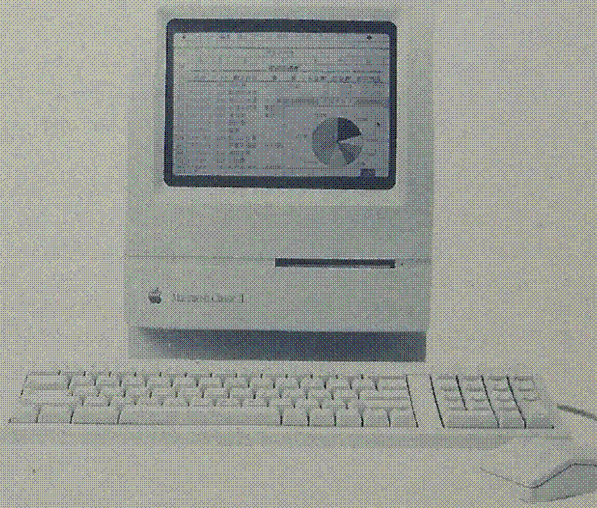


1名

■プロセッサ：80386SX（16 MHz）／メモリ：2MB／外部記憶装置：3.5インチFDD×1・40 MB HDD×1／ディスプレイ：16階調CFLバックライト式白黒液晶ディスプレイ

3

Macintosh ClassicII



1名

■プロセッサ：68030（15.667 MHz）／メモリ：4MB／外部記憶装置：3.5インチFDD×1／ディスプレイ：9インチビットマップ白黒ディスプレイ

7 RS-232C Library*3

(PC-9800用)

1名

提供：吉正電子㈱

8 Graphic Window Library*3

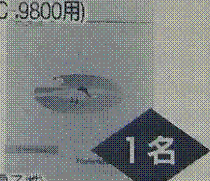
(PC-9800用)

1名

提供：吉正電子㈱

9 ISAM Library*3

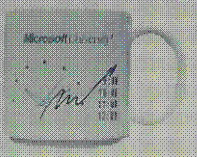
(PC-9800用)

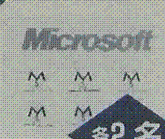


1名

提供：吉正電子㈱

13 マグカップ 14 Tシャツ 15 マウスパッド

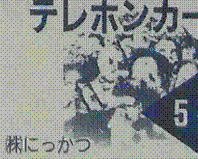




各2名

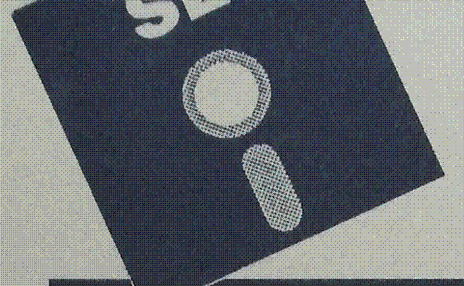
提供：マイクロソフト㈱

16 「ナイト・オブ・ザ・リビング・デッド」テレホンカード



5名

提供：㈱にっかつ



付録ディスクのご案内

このフロッピーディスクに含まれる実行ファイル，ソースリストおよびテキストファイルは「月刊Programmer's Page」1992年1月号に掲載されたものです．

付録のフロッピーディスクは，5"2HD（8セクタ／トラックMS-DOS）でフォーマットされています．3.5"や2DDなどへのコンバートサービスは行っておりません．なお，オンラインサービスSE-NET（03-5276-1297）でも，同じ内容のファイルを用意しています．

フロッピーディスクの内容は次の通りです．本誌およびディスクに含まれるドキュメントファイルをよく読んでからご利用ください．

■ディスクのおもな内容

●付録ディスクについての説明

ディレクトリ名：ルート

収録ファイル名：

README.BAT（ディスク内容の説明を読むためのバッチファイル，PC-9801専用）

README.DOC（ディスク内容の説明，MS-DOSテキストファイル）

ONLINE.BAT（オンラインソフトの説明を読むためのバッチファイル，PC-9801専用）

ONLINE.DOC（収録されているオンラインソフトの簡単な説明）

V.EXE（ミニページャー，PC-9801専用）

BOOK.TXT（翔泳社書籍目録）

●特集「プログラミングツール大集合」

①「開発の工程と道具としてのプログラム」（p.2）

ディレクトリ名：TOKU

収録ファイル名：TOKU.LZH

④「マクロの達人」（p.37）

ディレクトリ名：TOKU

収録ファイル名：

VZMAC.LZH（VZ用マクロ）

MIMAC.LZH（MIFES用マクロ）

BRMAC.LZH（Brief用マクロ）

●プログラム「遊」工房（p.80）

スクリーンセーバ

ディレクトリ名：PLAY

収録ファイル名：PLAY.LZH

●イカすBASICプログラム入門（p.85）

構造化がカッコイイ

ディレクトリ名：BASIC

収録ファイル名：QBASIC.LZH

●実践C塾（p.91）

ソート（並び替え）・その1

ディレクトリ名：JUKU

収録ファイル名：JUKU.LZH

●98活用アセンブラ技法（p.103）

SCSIデバイスドライバ／スケルトンの作成

ディレクトリ名：98ASM

収録ファイル名：98ASM.LZH

●綴じ込み付録

ディレクトリ名：DATA

収録ファイル名：ANSIC.TXT（MS-DOSテキストファイル）

●オンラインソフト採集帳（p.140）

本誌今月号のオンラインソフト採集帳で紹介したテキストファイル高速ページャー「MIEL」が収録されています．また「.LZH」という拡張子を持つ圧縮ファイルを復元するのに必要な，高圧縮復元書庫管理ソフト「LHA」，Cコンパイラ「LSI C-86 Ver.3.30試食版」が収録されています．それぞれのファイルはLHAによって圧縮されています．復元方法についてはルートにあるONLINE.DOCを参照してください．

ディレクトリ名：ONLINE

収録ファイル名：

MIEL237.LZH（MIEL v.2.37）

LHA213.EXE（LHA Ver.2.13，MS-DOS汎用）

LSIC330.LZH（LSI C-86 Ver.3.30試食版）

●クイズ・ランダムスクリーン（p.142）

クイズ解読ツール「PUZLE」

本誌掲載のクイズ・ランダムスクリーンの画像をディスプレイ上に再現し，マウスを使って並べ換えることにより，解答を得ることができるユーティリティです，PC-9801シリーズ専用です，詳しい使い方はPUZLE.DOCを参照してください．

ディレクトリ名：OMAKE

収録ファイル名：

PUZLE.EXE（解読ツール，PC-9801専用）

PUZLE.DOC（PUZLEの説明）

CONFIG.SYS（PUZLE用CONFIG.SYS）

AUTOEXEC.BAT（PUZLE用AUTOEXEC.BAT）

RNDSCREN.LZH（画像データ）

STBERY.LZH（12月号の解答データ）

STBERY.BMP（Windows 3.0の壁紙）

※使用上の注意

本誌に掲載されたプログラムおよびデータを利用するには，MS-DOSの基礎知識が必要です．このディスクには，MS-DOSのシステムは入っていません．別途，MS-DOSが必要となります．また，ソースファイルのコンパイルなど，別途コンパイラやユーティリティが必要な場合があります．

■圧縮ファイルについて

ディスクスペースを有効に使うために，多くのファイルは圧縮した上で収録しています．「.LZH」という拡張子のついたファイルは，高圧縮書庫管理ユーティリティ「LHA」によって圧縮されたファイルです．これらのファイルを利用するには，まずLHAで復元する必要があります．LHAの利用方法については，ONLINE.DOCをご覧ください．

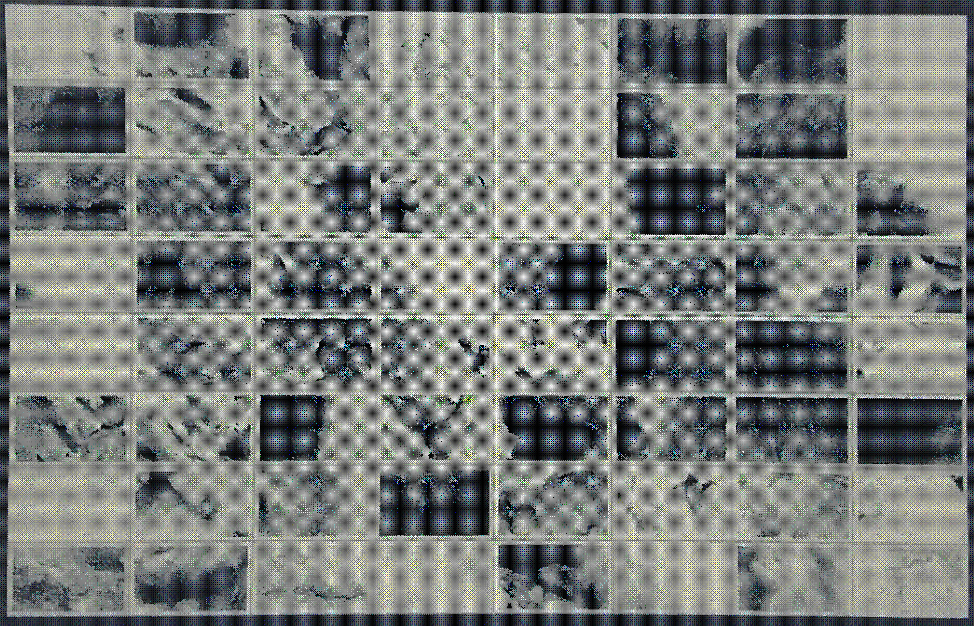
お詫び

本誌「実践C塾」のリスト5「WCSUB.C」（p.99～100）の中で，一部訂正がございました。本文中では訂正しておりますが，スケジュールの都合上，ディスク収録分については訂正分を反映させることができませんでした。訂正箇所は，p.100の91行～154行です。ディスク内のリストをお使いになる際には，お手数ですがこの箇所を修正の上，お使いいただけますようお願いいたします。

クイズ・ランダムスクリーン

ある１枚の画像を64分割し，ランダムに並べ替えてあります．
さて，もとの画像は何を写したものでしょうか．

ヒント：1992年は……



正解者の中から30名様に本誌表紙イラスト（日暮修一画）入り月刊プログラマーズ・ページ特製オリジナルマグカップをさしあげます．

応募方法：本誌に添付のアンケートはがきをご利用ください．

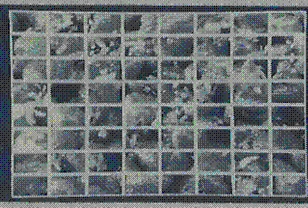
締め切り：1992年1月17日（消印有効）

なお，当選者の発表は発送をもって代えさせていただきます．

＊画像データは，付録ディスクやBBSでもサービスしています．

12月号の問題と正解

【問題】



【正解：いちご】

SE-NETからのお知らせ

SE-NETでは開設以来，オンラインサービスを行ってきましたが，回線混雑のため非常に接続しにくい等の問題が出てまいりましたので，GUESTによるダウンロードは中止させていただきます．今後は読者のみなさまと編集部とのコミュニケーションの場としてご利用ください．

●BBSへの接続

SE-NET　03-5276-1297

通信速度	300/1200/2400bps
接続プロトコル	CCITT V.22bis/V.22/V.21
エラー訂正プロトコル	CCITT V.42，MNPクラス5
漢字コード	シフトJIS

Programmer's Page
MONTHLY MAGAZINE FOR PROGRAMMERS

1991年1月号～1991年12月号

月刊プログラマーズ・ページ年間総目次

■1991年1月号

特集　プログラマの仕事場
徹底レポート「プログラマの仕事場」

プログラマの生態学　コンピュータの登場とプログラマ社会の形成　20
Windows，プレゼンテーションマネージャ関連のソフトウェアを手がけるプログラマーズカンパニー　イースト株式会社　26
制御系のハードウェア設計からその制御ソフトまでこなすからCADとCコンパイラが必需品　平田祥司(株式会社マイクロフト)　38
業界トップの開発環境は国際ネットワークに接続されたハイパワーシステム　マイクロソフト株式会社　46
より広いコンピュータの世界と快適な開発環境を求め続けるアクティブなプログラマの生活とは　西川和久(フリープログラマ)　52
Windowsのパッケージソフト参入で一躍注目を浴びはじめた古参ソフトハウス　ジィスクソフトウェア株式会社　58
道具としてのソフトウェアと情報としてのソフトウェアを考えながらマッキントッシュ中心の開発を進める　株式会社演算星組　64
ユニークな開発支援ツールを生みだす開発環境はシンプル・イズ・ザ・ベスト　メガソフト株式会社　70
Turbo Cシリーズは優れた製品であるとともにボーランド社自身の開発ツールでもある　株式会社ボーランドジャパン　76
ヒットメーカーの原動力はインテリジェントビルに集う120人のプログラマ　株式会社ジャストシステム　82
LANを導入した開発環境にはパソコンソフトウェア開発の創造性が息づいている　株式会社大塚システム研究所　86
ハードの性能を生かすアセンブラによる開発システム　押入にはハードウェア工房も　小高輝真(フリープログラマ)　92

実践C塾　第1回
MS-DOS上のC処理系の現状　水居浄　102
98活用アセンブラ技法／ビデオシステム編　第1回
MASM・TASMによる構造化マクロライブラリの製作　星野操　114
プログラマのためのハードウェア再入門1
シリアルインターフェースRS-232C　鎌田正寿　126
プログラム「遊」工房　壱乃巻
テロップ表示プログラム　谷口典生　130
Macintoshのプログラミング環境1
Macintoshのプログラムとはどういうものだろうか　大座畑重光　136
今月の受験情報1
情報処理技術者になるメリットとは　金丸秀樹　140

パソコン博物館　第1回
米国での曙／ALTAIR 8800　井戸禎光　17
プログラマの道具学　第1回
鉛筆　祐安重夫　96
オダジマタカシのただいま悪戦苦闘中
ベーシックの夜は更けて　小田嶋隆　140

INFORMATION PAGE 1　1
NEWS & NEW PRODUCTS　2
データショウ'90開催，ノート・ブック型全盛，さらに小型化と差別化への動き
COMDEX/Fall'90開催，出展社数，入場者数ともニューレコードを達成！
ボーランドジャパン，PC-9801版TURBO C++コンパイラを発売
マイクロソフト，MS-C Ver.6.0を発売
ライフボート，C言語用画面作成ツールライブラリを発売
吉正電子，MS-C 6.0およびQuickC 2.0対応の開発支援ライブラリを発売
富士通，FM TOWNSの新機種を発売
日本IBM，PS/55の新モデル4機種ほかを発売
富士通，ノートブック型パソコンFM R-CARDを発売
アップルコンピュータジャパン，Macintoshの新モデルを発売
日本電気，A5ファイルサイズのハンディパソコンを発売
エプソン，32ビットマシン4機種を発売
三菱電機，32ビットノート型AXマシンを発表
東芝，DynaBookの新機種を発売
ASTリサーチ・ジャパン，PC/AT・PC-9801互換マシンを発売
ソニー，RISC NEWSシリーズ最上位機を発表
東芝，40MHzの32ビットRISCプロセッサを搭載したWSを発売
富士通，「Sファミリーエンジニアリングワークステーション」ラインナップ強化
コンテック，PC-9801N/NS用のI/Oユニットを開発
エプソン，日本語ページプリンタの新機種を発売
アイ・オー・データ機器，All in Note用増設メモリボード発売
日本IBM，IBM DOSバージョンJ4.0/Vを発売
ライフボート，DPMI委員会に参加
デジタル・リサーチ・ジャパン，日本語DR-DOS 5.0を販売開始
佐々木哲のニューヨークシステム通信　12
西川和久の希望的観測　14
NEXT STAGE　16

INFORMATION PAGE 2　145
Book Review「Life with UNIX」中村正三郎　146
本のアンテナ　古田島義和
Programmer's Program「日本語Brief」　148
プログラマ登場①　園田康博　149
オンラインソフト採集帳1「スピード」　150
HEALTHY? UNHEALTHY?①　東京駅にはドリンク剤のバーがある
コンピュータの法律「プログラムの画像に著作権は認められるか」
熊谷信太郎　152
アキバ短信「ラオックス　ザ・コンピュータ館」
バーチャルスペースガイド　154
創刊記念プレゼント　156
ドラゴン・ポスト
クイズ・ランダムスクリーン　158

綴じ込み付録
MIFES，VZ-Editor標準キー設定，機能番号比較一覧表

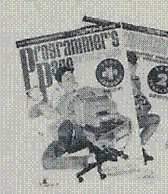
■1991年2月号

特集　Windowsの時代である！
Windowsプログラミングを徹底解説・大解剖

窓(ウィンドウ)の歴史　20
Windows 3.0登場
Windows 3.0がパソコン環境に新しい展開をもたらす　26
Windowsプログラミング講座 1
Windowsシステムとプログラムの条件　36
Windowsプログラミング講座 2
Windows 3.0＋MS-C 6.0による開発環境構築法　永井知己　44
Microsoft Windows Device Development Kit(DDK)　西川和久　52
Windowsプログラミング講座 3
Windows基本プログラム大解剖　永井知己　54
Windowsプログラミング講座 4
実践的Windowsプログラミング技法　滝口力　69
Windowsプログラミング講座 5
新しいWindowsプログラミング
WindowsMAKER ver.3.01　西川和久／Asymetrix ToolBook／Actor 3.0　79

Macintoshで考えたこと
Personalからintimateへ　大座畑重光　94
実践C塾　第2回
メモリモデルとオーバーレイ　水居浄　98
98活用アセンブラ技法／ビデオシステム編　第2回
半角外字フォントの読み込みと設定　星野操　112
プログラマのためのハードウェア再入門 2
ディスプレイ　鎌田正寿　122
プログラム「遊」工房　武乃巻
モアレ模様　谷口典生　127
今月の受験情報
情報処理技術者試験の動向　赤松高文　133

パソコン博物館　第2回
システムとして売り出されたパソコン／APPLE II　井戸禎光　17
プログラマの道具学　第2回
電卓　祐安重夫　90
オダジマタカシのただいま悪戦苦闘中
小田嶋隆　92



綴じ込み付録
MS-Windows 3.0関数一覧

INFORMATION PAGE1　1
NEWS & NEW PRODUCTS　2
日本IBM，必要メモリを抑えたOS/2 1.3を発表，SEは2MB，EEは3MBで動作
日本電気，16MHzのV30を搭載した98NOTEの後継機などPC-9800シリーズの新モデル5種を発売
富士通，486を搭載したFM Rシリーズの最上位機種を発売
ソニー，NEWSの新機種を発売，ラップトップWSでは最高速の17MIPSを実現
東芝，DynaBooK386シリーズとTFTカラー液晶のラップトップ型パソコンを発売
日本IBM，光磁気ディスクが内蔵可能なPS/55Zの新モデルを発売
日本IBM，ハードディスクを拡張した5530Tの新モデル3種を発売
日本電気，ネットワーク型教育システムPC SEMI Kシリーズを発売
富士通，教育分野向けに機能強化したFM TOWNS新モデル2種を発売
日本IBM，高解像度XGA対応のPS/2最上位モデルなどを発売
エプソン，V.42-MNPクラス5搭載のパーソナルモデムを発売
日本電気，98NOTEなどに対応の4MB，8MB用増設メモリを発売
アイ・オー・データ機器，55Z用SIMメモリなどを発売，増設メモリのラインナップを強化
メルコ，DRAMを用いたシリコンディスクを発売
日本IBM，ワークステーション用最高速プリンタを発売
イースト，Windows 3.0対応の日本語ワープロソフトを発表，オンラインヘルプやカラーアイコンに対応
日本電気，日本語OpenDeskなどPC-9800用UNIX関連ソフトを販売開始
マイクロソフト，Microsoft WorksのDynaBook版を発売
ツァイト，Z's STAFF KiD98 Ver.3.0を発売
マイクロソフトとキヤノン販売がMacintosh用ソフトを共同開発
アスキー，インフォミックス社と提携強化
マイクロソフト，片貝システム研究所とジョイントし公認トレーナー制度「MOT」を開始
アスキーネットPCS，9600bpsの高速通信サービスを開始，従来より8倍高速な情報サービスを提供
OS-9000の開発パッケージがVer.1.1にバージョンアップ，オプショナルパッケージも提供開始
佐々木哲のニューヨークシステム通信　12
西川和久の希望的観測　14
NEXT STAGE　16

INFORMATION PAGE2　137
Book Review「コンパイラ I，II」　中村正三郎　138
本のアンテナ　古田島義和
Programmer's Program「RCS」　140
プログラマ登場②　片桐明　141
オンラインソフト採集帳 2「FD」　142
HEALTHY? UNHEALTHY?②プログラマもカゼをひく，カゼをひいたら……
コンピュータの法律「プログラム著作権とはどんなものか－その1」
熊谷信太郎　144
アキバ短信「古炉奈」
バーチャルスペースガイド　146
創刊記念プレゼント　148
ドラゴン・ポスト
クイズ・ランダムスクリーン　150

■1991年3月号

特集　この言語を使いなさい！！

パソコン・プログラミング言語盛衰記　11
プログラミング言語の仲間たち
その誕生から系譜を辿る　神林靖　18
Assembler
TSR，デバドラなどに使われるDOSの主力言語　小高輝真　28
処理系紹介　MASM 5.1／Turbo Assembler／OPTASM／マシン語マスター98／Advanced RUN/MASM
BASIC
すぐに乗れる自転車のような手軽さ　山本英昭　33
処理系紹介　MS-QuickBASIC／BASIC/98Pro／N88-BASIC
C
考えたことを素直に記述できる　辻村宏　37
処理系紹介　Turbo C／MS-C／MS-QuickC／Lattice C／LSI C-86／Whitesmith C／NDP C386／HIGH C386／Power C／Advanced RUN/C／C-terp／Let's C／Visual C／ORCHID C
C＋＋
オブジェクト指向を取り入れた次世代C言語　大野元久　43
処理系紹介　Turbo C＋＋／IDEA C＋＋／MIWA C＋＋／Zortech C＋＋
COBOL
大型機の経験を生かせる互換機　海老原浩之　46
処理系紹介　Advanced RUN/COBOL／日本語Master COBOL-74／日本語LEVEL II COBOLコンパイラ／日本語COBOL/2コンパイラ／日本語PC-COBOL
FORTRAN
科学技術計算の世界共通語　桜井健次　50
処理系紹介　MS-FORTRAN／Pro FORTRAN-77／SW-Pro FORTRAN-77／Advanced RUN/FORTRAN／AWS-FORTRAN／RM/FORTRAN／NDP FORTRAN386
HyperCard/HyperTalk
高度なソフトを容易に制作できる　田村浩二　55
Lisp
高速・大容量化時代に能力を発揮する　藤沢滋　58
処理系紹介　ICLisp／Hyper Lisp／GCLISP／It's CL/386／Waltz LISP／Advanced RUN/LISP／muLISP87／XLISP／XSCHEME
LOGO
効率より学習の容易さを優先　福田洋　63
処理系紹介　ロゴライター／3D-LOGO
Mind
日本語プログラミング言語　加野島英渡　67
処理系紹介　Mind 5.1
Modula-2
モジュール化を重視した文法　西端芳彦　71
処理系紹介　FTL MODULA-2／LOGITECH MODULA-2 Ver.3.0
Pascal
ミスを適切に指摘してくれる優しさ　オケゲム　76
処理系紹介　Turbo Pascal／Pro PASCAL／Advanced RUN/PASCAL

Prolog
強力なプログラムを速く記述する　鈴木太朗　80
処理系紹介　IF/Prolog Ver.3.4／Arity Prolog V5／Advanced RUN/PROLOG／Prolog-KABA／Astral Prolog／Prolog-J
Smalltalk
柔軟性のあるオブジェクト指向言語　近江武一　84
処理系紹介　Smalltalk/v／Smalltalk-80 System Package／Smalltalk-80/VM68版

実践C塾　第3回
MS-CとTurbo Cのコード生成の特徴　水居浄　94
98活用アセンブラ技法／ビデオシステム編　第3回
フォント作成プログラム　星野操　104
プログラマのためのハードウェア再入門3
セントロニクス　鎌田正寿　114
プログラム「遊」工房　参乃巻
迷路作成プログラム　谷口典生　118
今月の受験情報
自ら動機づけをして，まず2種から挑戦　赤松高文　125

パソコン博物館　第3回
ゲームマシンの先駆けとなった電子帳簿／PET2001　井戸禎光　9
プログラマの道具学　第3回
モデム　祐安重夫　90

綴じ込み付録
Microsoft C 6.0関連プログラム起動オプション

INFORMATION PAGE1　1
NEWS & NEW PRODUCTS　2
マイクロソフトウェア，Zortech C++の独占契約を締結，286版はWin 3.0にも対応
MS-Windows 3.0日本語版がリリース
コンテック，自社ボードをサポートしたQB用ドライバソフト発売
日本電気，386を搭載したデスクトップ機など4機種16モデルを発売
マイクロソフト，Windows 3.0対応のExcelを発売
クレオ，グラフィック，通信機能を備えた低価格ワープロソフトを発売
コンテック，EMS対応増設メモリボードを発売
アメリカン・テクノロジー・グループ，超高速モデムを普及価格で発売
アイ・オー・データ機器，98拡張用スロットを応用した新規格を発表
佐々木哲のニューヨークシステム通信　4
西川和久の希望的観測　6
Programmer's Rest Bench　8

INFORMATION PAGE2　129
Book Review「The Annotated C++ Reference Manual」
中村正三郎　130
本のアンテナ　古田島義和
Programmer's Program「IOS-10STD」　132
プログラマ登場③　兵藤嘉彦　133
オンラインソフト採集帳3「WTERM Ver.8.20」　134
HEALTHY? UNHEALTHY?③ポカポカ，スベスベ，入浴剤は効き目？気分？
コンピュータの法律「プログラム著作権とはどんなものか－その2」
熊谷信太郎　136
アキバ短信「若松通商」
バーチャルスペースガイド　138
創刊記念プレゼント　140
ドラゴン・ポスト
クイズ・ランダムスクリーン　142

■1991年4月号

特集　C読本
C言語と正しく付き合うためのポイント

1　Cを知らないあなたのための
体で覚えるC言語入門　種村明宏　2
2　ワークステーションからパソコンへ
UNIXのC，MS-DOSのC　辻村浩　15
3　読みやすいプログラムとは？
Cプログラミングのスタイルを決める　秋山登　25
4　うまくいかないこともある！？
C言語のデバッグテクニック　平賀礼二　37
5　C言語の明日
オブジェクト指向Cプログラミング　石川雅基・神林靖　47
C言語関連書籍一覧　61

プログラム「遊」工房　四乃巻
「マスターマインド」ゲーム　谷口典生　70
MS-DOSツール・ラボ　第1回
ディレクトリ表示ツール L.EXE　加野島英渡　76
実践C塾　第4回
コードヘルパと浮動小数点演算　水居浄　84
プログラマのためのハードウェア再入門4
フロッピーディスク　鎌田正寿　94
98活用アセンブラ技法／ビデオシステム編　第4回
アセンブラと高級言語のリンク・および常駐サブルーチン
星野操　98

パソコン博物館　第4回
Z80成功物語の嚆矢／TRS80　井戸禎光　66
プログラマの道具学　第4回
キーボード　祐安重夫　68
情報処理技術者試験 第2種合格！トレーニング
第1回　ハードウェアの基礎！　勝島貴之　113
Book Review「オブジェクト指向入門」　中村正三郎　118
本のアンテナ　古田島義和
佐々木哲のニューヨークシステム通信　120
西川和久の希望的観測　122
コンピュータの法律「市販のプログラムのコピーは自由か」
熊谷信太郎　124
アキバ短信「パソコンソフトウェアラボラトリ」
プログラマの仕事場　第1回　株式会社ソフトヴィジョン　126

INFORMATION PAGE　129
NEWS & NEW PRODUCTS　130
UNIXインターナショナル，1991年版ロードマップを発表
東芝J-3100にデスクトップタイプ登場，486搭載，EISAバス対応モデルも
エプソン，拡張性を重視した32ビットブック型パソコンを発表
日立，68040搭載のワークステーション2機種を発表
日本電気，N5200シリーズにカラーラップトップなどの新機種を発表
エプソン，200MBの高速大容量SCSI対応ターミナルハードディスクを発売
アイ・オー・データ機器，98セカンドバス規格対応のEMSプラス増設RAMボードを発売
メモレックス・テレックス，PS/55用Netware対応ハードディスクを発売
ソフトウェアジャパン，日本語版ピーター・ノートン・コマンダーVer3.0を発売
バックス，Windows 3.0に対応したVJE-γを発売
三菱商事，SQLBase，SQLWindowsの日本語版リリースを発表
アスキー，マルチメディア対応のインフォミックスversion4を発売
アイ・オー・データ機器など4社，アプリケーションの組み込み規格MAOIX（マオイクス）を提唱
沖電気工業，日本初のフューチャーバス＋採用を発表
Programmer's Program「QuickHelp」　134
オンラインソフト採集帳4「LHa Ver.2.05」　136
HEALTHY? UNHEALTHY?④食生活の片寄りは仕事にも影響する？
SE-NET拾い読み　138
ドラゴン・ポスト　140
バーチャルスペースガイド　141
読者プレゼント
付録ディスクのご案内　142
クイズ・ランダムスクリーン　143
オンラインサービス（SE-NET）のお知らせ

付録　5"2HDフロッピーディスク
綴じ込み付録
Turboシリーズ　プログラム起動オプション

■1991年5月号

特集 基礎力としてのアセンブラ
コンピュータを理解し，その能力をひきだすために

1 いまどきのアセンブラプログラミング入門 秋山登　2
アセンブラを必要とするこれだけの理由
2 CPUアーキテクチャとマシン語 宮川誠一　17
86系からRISCまでを概観する
3 高級言語とアセンブラのプログラミング 加野島英渡　37
リンク＆インラインアセンブラ
4 逆アセンブルの楽しみ 清十郎　51
ツールとその使い方

プログラム「遊」工房 伍乃巻
「mine sweeper」ゲーム 谷口典生　72
MS-DOSツール・ラボ 第2回
割り込みベクタ表示・設定プログラムVECT.EXE 加野島英渡　78
プログラマのためのハードウェア再入門5
仮想記憶 鎌田正寿　89
98活用アセンブラ技法／ビデオシステム編 第5回
バックカラー処理 星野操　93
実践C塾 第5回
プロセス制御／システムコールライブラリ 水居浄　104

パソコン博物館 第5回
日本におけるパソコンの萌芽／TK-80 井戸禎光　68
プログラマの道具学 第5回
猫 祐安重夫　70
Book Review「Undocumented DOS」 中村正三郎　118
情報処理技術者試験 第2種合格！トレーニング
第2回 ハードウェアの基礎2 勝島貴之　113
本のアンテナ 古田島義和
佐々木哲のニューヨークシステム通信　120
西川和久の希望的観測　122
コンピュータの法律「バックアップ用のコピーに関する問題点」
熊谷信太郎　124
アキバ短信「ダニエル」
プログラマの仕事場 第2回
株式会社ミワシステムズコンサルティング　126

付録 5"2HDフロッピーディスク
綴じ込み付録
8086インストラクション一覧

INFORMATION PAGE　129
NEWS & NEW PRODUCTS　130
ボーランドジャパン，PC-9801版Turbo Pascal 6.0を発売
マイクロソフト，MS-C 6.0用開発支援ツールパッケージを発売
レンガ，C言語ファイル圧縮ライブラリを発売
オオツカ商事，C言語開発支援ツールCPAD Ver.2.0を発売
FAシステムエンジニアリング，C言語用開発支援ライブラリ2種を発売
オーキッド，効率的な環境を構築するC言語用開発ツールを発売
マイクロフォーカスジャパン，if386 AXなどに対応するCOBOL/2の供給を開始
エプソン，3.5インチFDD2基内蔵の32ビットマシンを発売
日本IBM，日本語Windows V3.0をインストールしたPS/55Zの新モデルを発売
日本IBM，DOS/Vに対応した初のノート型パソコンを発売
富士ゼロックス，ワークステーションの新機種を発売
日本エヌ・シー・アール，ワークステーションの新機種を発売
アップルコンピュータジャパン，レーザー品質のプリンタ2種を発売
富士通，全二重・半二重の通信が可能な複合モデムを発売
日本IBM，PS/55用プリンタの新機種を発売
エプソン，IBMパソコン対応のカラー漢字プリンタを発売
アイ・オー・データ機器，PC-9801DA用RAMボードを発売
富士通，高精細表示ディスプレイを発売
日本IBM，高解像度とVGAに対応した日本語Windows V3.0を発売
アップルコンピュータジャパン，日本語版および英語版HyperCard 2.0を発売
日本サン・マイクロシステムズ，日本語OpenWindowsの本格出荷を開始
シンフォニー，MS-Windows 3.0対応高機能チャートエディタを発売
ボーランドジャパン，RDBソフトPC-9801版「PARADOX 3.5」を発売
富士通，NetWare386を発売．製品は10・40・100クライアント用の3種
パーソナルメディア，カード型DB「Let'sアイリス」UNIX版を発売
ザザンパシフィック，英文文章解析ツールの98対応版を発売
日立製作所，超高密度磁気記録技術を開発．3.5"で10GBのHDDも可能に
ソニー，従来の6倍の密度をもつ光信号記録再生技術を開発
マイクロソフトの「Pen Windows」に21社が採用の意向を発表
「日本語MS LANマネージャ共通規約」が策定
富士通，複数チップを1つのパッケージに搭載する技術を開発
日本IBM，PCオープンアーキテクチャー推進協議会（OADG）を設立
シグマシステムとX/Open，OSF，UIとの協調関係確立
Programmer's Program「PPII」　136
オンラインソフト採集帳5「NUKEM」　138
HEALTHY? UNHEALTHY?⑤機能性食品はほんとうに機能する？
ドラゴン・ポスト　140
バーチャルスペースガイド　141
読者プレゼント
付録ディスクのご案内　142
クイズ・ランダムスクリーン　143
オンラインサービス（SE-NET）のお知らせ

■1991年6月号

特集 研究「プログラミング統合環境」
快適で効率的な開発環境を考える

プログラム作成支援環境の動向　2
用途により言語と支援環境の形態は異なる 鈴木則久
言語製品の流れとプログラミング環境の発展　7
PWB/Quick/IDE　10
PC-9801用Cコンパイラの統合環境を比較する
PWBは標準ツールとなり得るか？　18
MS-C 6.0の統合開発環境
メモリを気にせず使えるOS/2上のPWB 海老原浩之
PWBのカスタマイズ つけ加えたい機能をCでプログラムする
種村明宏
計画的に使うことで効率を高める　33
MS-DOSの開発環境／QuickCの使い方 小松伸行
●マイクロソフトに聞く プロ指向とQuickシリーズの2本立てで展開
プロフェッショナル向けに新たな機能強化　45
Turbo C++/Borland C++のIDE 大野元久
●ボーランドジャパンに聞く Windowsが製品のキーワードになった
人にやさしいプログラミング環境　57
Macintoshの統合環境とプログラミング 新居雅行
これからの統合環境 Windows上の統合環境プログラミング　66
Turbo Pascal for Windows
Smalltalk/V for Windows
Actor 3.0
統合環境を備えた言語製品紹介
Microsoft C Professional Development System Version 6.0
MS-QuickC 2.0
MS-QuickBASIC 4.5
Turbo C++
Turbo Pascal 6.0
Smalltalk/V
THINK C
THINK Pascal

プログラム「遊」工房 六乃巻
「タイピング練習」ゲーム 谷口典生　76
実践C塾 第6回
システムコールライブラリ／日本語処理ライブラリの比較 水居浄　82
プログラマのためのハードウェア再入門6
ハードウェアのゆくえ 鎌田正寿　88
MS-DOSツール・ラボ 第3回
再帰と式の評価 簡易16進電卓EXPR.EXE 加野島英渡　92
98活用アセンブラ技法／ビデオシステム編 第6回
バックカラー付き画面表示ルーチンの作成 星野操　101

パソコン博物館 第6回
ベンチャービジネス，希望の星／SORD Mシリーズ 井戸禎光　72
プログラマの道具学 第6回
ねずみ 祐安重夫　74
情報処理技術者試験 第2種合格！トレーニング
第3回 ソフトウェアの基礎II 長澤洋一郎　113
Programmer's Rest Bench　117
Book Review「コンサルタントの秘密」 中村正三郎　118
本のアンテナ 古田島義和
佐々木哲のニューヨークシステム通信　120
西川和久の希望的観測　122

コンピュータの法律「私的使用のためのプログラムのコピー」
熊谷信太郎　124
アキバ短信「T-ZONE」
プログラマの仕事場　第3回　株式会社エム・ビー・テクノロジー　126

付録　5"2HDフロッピーディスク
綴じ込み付録
統合開発環境メニュー一覧

INFORMATION PAGE　129
NEWS & NEW PRODUCTS　130
シンフォニー，Windows 3.0対応の統合開発環境「CASE:W」を発売
ライフボート，高速科学技術計算パッケージMathAdvantageを発売
メディカルリサーチイクイップメント，高速行列演算コンパイラを発売
関西電機，486マシンを含むAT互換機4種を発売
三菱電機，クライアントサーバコンピュータ「apricotシリーズ」を発売
日本IBM，POWERステーションの最高速モデルなど2機種を発売
コンテック，NetWare対応LANアダプタシリーズ34種を発売
日本IBM，ISDN対応などのネットワーク関連製品を発表
マイクロソフトウェア，SUN上でDOS用ソフトを実行するツールを発売
レンガ，Windows 3.0対応アプリケーション2種を発売
マイクロソフト社など21社が先進的プラットフォームACE構想を提唱
AX協議会，VGA機能をもつビデオシステムの仕様を決定
Programmer's Program「Microsoft Developer's Toolkit」　134
オンラインソフト採集帳6「BDIFF/BUPDATE」　136
HEALTHY? UNHEALTHY?⑥「オタクデブ」にさようなら
SE-NET拾い読み　138
ドラゴン・ポスト　139
付録ディスクのご案内　140
バーチャルスペースガイド　141
読者プレゼント
クイズ・ランダムスクリーン　142
オンラインサービス（SE-NET）のお知らせ
バックナンバーのお知らせ　143

■1991年7月号

特集　最新「デバイスドライバ」研究

MS-DOSデバイスドライバの技法と新しいOS環境における基礎

入出力装置とのインターフェースを確立する
デバイスドライバ入門　日笠健　2
プログラムの基本構造は共通
MS-DOSデバイスドライバの書き方　虎岩登　19
アセンブラが苦手でもデバイスドライバは書ける
C言語によるデバイスドライバ作成法　秋山登　26
その実体はダイナミックリンクライブラリ
Windowsのデバイスドライバ　37
マルチタスク環境下のデバイス制御
OS/2のデバイスドライバ　46
PRODUCTS REVIEW
MS-C 6.0に続くプロ指向の言語パッケージ
Microsoft BASIC Professional Development System Ver.7.1
海老原浩之　58
プログラム「遊」工房　七乃巻
「三子棋（サン・ツー・チ）」ゲーム　谷口典生　64
IC千夜一夜　第1回
IC技術の基礎知識　虎岩登　74
MS-DOSツール・ラボ　第4回
ファイル検索プログラムT.EXE　加野島英渡　78
98活用アセンブラ技法／ビデオシステム編　第7回
ウィンドウシステムの作成　星野操　90
実践C塾　基礎テクニック篇　第1回
引数／オプションの解析　大野元久　102
付録　5"2HDフロッピーディスク
綴じ込み付録
MS-DOS Version 3.3 ファンクションリクエスト一覧

パソコン博物館　第7回
世界を制覇したパソコン／IBM PC　井戸禎光　70
プログラマの道具学　第7回
酒　祐安重夫　72
情報処理技術者試験 第2種合格！トレーニング
第4回　ソフトウェアの基礎II　長澤洋一郎　113
Book Review「An Introduction to TEXT PROCESSING」
中村正三郎　118
本のアンテナ　古田島義和
佐々木哲のニューヨークシステム通信　120
西川和久の希望的観測　122
コンピュータの法律「プログラムの改造・その1」
熊谷信太郎　124
アキバ短信「秋月電子」
プログラマの仕事場　第4回　株式会社バックス　126
INFORMATION PAGE　129
NEWS & NEW PRODUCTS　130
第72回ビジネスショウ開催される．Windows 3.0，486マシン，DOS/V……
アップルコンピュータジャパン，次世代OS「System 7」を発売
日本電気，486SX搭載マシンなどPC-9800シリーズの新製品を発表
日本IBM，OADG標準機となる低価格のDOS/V用パソコンを発売
セガ・エンタープライゼス，2CPUパソコン「テラドライブ」を発売
日立，VGA搭載のAXマシン「FLORA」シリーズを発売
サザンパシフィック，dBXL，dBXL/LAN，QuickSilverをバージョンアップ
Programmer's Program「GENIFA」　134
オンラインソフト採集帳7「DCP」　136
HEALTHY? UNHEALTHY?⑦「食品成分表」の効用
ドラゴン・ポスト　138
付録ディスクのご案内　140
バーチャルスペースガイド　141
読者プレゼント
クイズ・ランダムスクリーン　142
オンラインサービス（SE-NET）のお知らせ
バックナンバーのご案内　143

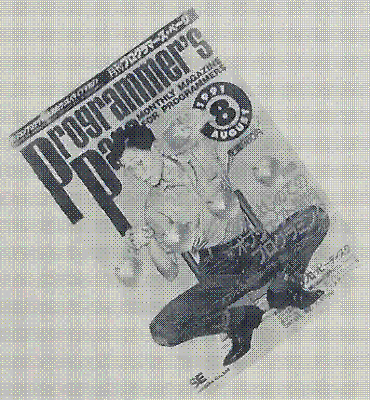
■1991年8月号

特集　はじめてのオブジェクト指向プログラミング

オブジェクト指向な…分析・設計，プログラミングとは？

オブジェクト指向のパラダイムとプログラミング言語の実際
オブジェクト指向概念の基礎　神林靖　2
オブジェクト指向プログラミングの用語解説　神林靖　11
柔軟で効率的なシステムを構築するために
オブジェクト指向による分析・設計の重要性
アリス・サルペーター　15
よりよいCからオブジェクト指向へ
C++の現状と未来　阿部伸　23
システムとプログラム開発は分けて考える
Windowsにおけるオブジェクト指向とは？　34
クラスライブラリとの格闘はまぬがれない
Macintosh環境におけるオブジェクト指向　新居雅行　43
オブジェクト指向ライブラリTurbo Visionの実体
Turbo Pascal 6.0に見るオブジェクト指向　50
もっと詳しくオブジェクト指向プログラミングを学びたい人に　60

プログラム「遊」工房　八乃巻
「サイクロイド／トロコイド／スピログラフ描画」プログラム
谷口典生　62
IC千夜一夜　第２回
４ビットマイクロプロセッサの登場－4004と4040－　虎岩登　72
MS-DOSツール・ラボ　第５回
スタックとFORTH①FORTHコンパイラFAST　加野島英渡　76
98活用アセンブラ技法／ビデオシステム編　第８回
バックカラー表示常駐プログラムの作成　星野操　88
実践Ｃ塾　基礎テクニック篇　第２回
文字列の入力（前編）　大野元久　101



付録　5"2HDフロッピーディスク

綴じ込み付録

C++文法サマリ

パソコン博物館　第８回
オールインワン・パソコン／OKI if800　井戸禎光　68
プログラマの道具学　第８回
日本語ワードプロセッサ　祐安重夫　70
情報処理技術者試験 第２種合格！トレーニング
第５回　工業　小宮正好　113
Book Review「Ｃ言語による最新アルゴリズム事典」
中村正三郎　118
本のアンテナ　古田島義和
佐々木哲のニューヨークシステム通信　120
西川和久の希望的観測　122
コンピュータの法律「プログラムの改造・その２」
熊谷信太郎　124
アキバ短信「まつや」
プログラマの仕事場　第５回　株式会社アドミラルシステム　126

INFORMATION PAGE　129
NEWS & NEW PRODUCTS　130
　ボーランドジャパン，Winアプリ開発が可能な「BORLAND C++ 2.0」を発売
　マイクロソフト，DOS/V初の言語製品PS/55用MS-C 6.0を発売
　ライフボート，4GBのメモリ空間を使用できるDOSエクステンダーを発売
　吉正電子，同期式をサポートしたRS-232Cライブラリを発売
　エプソン，ハイレゾモード対応の低価格32ビットマシンを発売
　東芝，386SX/20MHz，プラズマディスプレイ搭載のDynaBookを発売
　ソニー，書き換え可能な3.5インチ光磁気ディスクシステムを商品化
　インテルジャパン，NetWare386J対応のプリントサーバを発表
　コンテック，小規模パソコンLAN用ネットワークOSを発売
　富士通，TOWNSのマルチメディア機能を大幅強化する製品群を発売
　メルコ，98DA，RA用内部増設RAMボードを低価格で発売
　インテルジャパン，50MHzの486マイクロプロセッサを発表
Programmer's Program「SOURCER」　134
オンラインソフト採集帳８「CINIT.SYS」　136
HEALTHY? UNHEALTHY?⑧米国で提唱された新しい食品分類
ドラゴン・ポスト　138
付録ディスクのご案内　140
バーチャルスペースガイド　141
読者プレゼント
クイズ・ランダムスクリーン　142
オンラインサービス（SE-NET）のお知らせ
バックナンバーのご案内　143

■1991年９月号

特集　グラフィックスプログラミングの世界

基礎からレイトレーシングまで，プログラミング技法を徹底解説

グラフィックスといってもいろいろあるけれど……
グラフィックスプログラミング入門　小林龍正　2
1677万色・フルカラーグラフィックスプログラミング
フレームバッファ活用法　尾崎行雄　14
超リアルな画像作成テクニック
レイトレーシング入門　稲蔭正彦　26
DOS/Vで注目される世界標準機のグラフィックス
IBM PCのビデオシステム入門　柳澤健二　36
ハードウェアに依存しないグラフィックス
Windowsにおけるグラフィックスプログラミングの実際
中新田秀美　47
市販フレームバッファ・グラフィックスライブラリ製品紹介　63
　HyPER-FRAME＋
　写像
　SuperFrame²/SuperFrame²Σ
　Bit Map Text & Graphics Adaptor Ver.1.0
　GENIFA
　UCG（Ultra C Graph）Ver.2.0
　Graphic & Mouse（/EX）

パソコン博物館　第９回
インテリジェントペリフェラルコントローラ／ベーシックマスターレベル３　井戸禎光　66
プログラマの道具学　第９回
シンセサイザ　祐安重夫　68
情報処理技術者試験 第２種合格！トレーニング
第６回　商業　佐々木秀夫　113
Book Review「続ソフトウェア千夜一夜物語」　中村正三郎　118
本のアンテナ　古田島義和
佐々木哲のニューヨークシステム通信　120
西川和久の希望的観測　122
コンピュータの法律「プログラムの改造・その３」
熊谷信太郎　124
アキバ短信「コムサテライト」
プログラマの仕事場　第６回　アンテナハウス株式会社　126

プログラム「遊」工房　九乃巻
「射撃」ゲーム　谷口典生　70
IC千夜一夜　第３回
８ビットマイクロプロセッサの発展－8008と8080－　虎岩登　76
MS-DOSツール・ラボ　第６回
コンパイラの自己記述　FORTHコンパイラ②FAST　加野島英渡　80
実践Ｃ塾　基礎テクニック篇　第３回
文字列の入力（後編）　大野元久　91
98活用アセンブラ技法／番外編　第１回
アセンブラ命令コードの把握　星野操　103

INFORMATION PAGE　129
NEWS & NEW PRODUCTS　130
　日本IBM，大容量メモリに対応した「DOS 5.0」を発表
　テックソフト＆サービス，4GBのメモリが使えるHIGH C386をバージョンアップ
　マイクロウェア・システムズ，CD-Iタイトル開発ツールを発売
　東芝，大容量HDD内蔵の32ビットデスクトップパソコンを発売
　日本電気，カラー液晶搭載の32ビット低価格ラップトップマシンを発売
　三菱電機，３モードFDDを搭載した32ビットノートブック型パソコンを発売
　アイ・オー・データ機器，98対応シリコンディスクボードほかを発売
　アイシーエム，98NOTE対応のハードディスクパックを発売
　エプソン，マッキントッシュ対応プリンタドライバソフトを発売
　インターコム，通信ソフト「まいとーく」をバージョンアップ
　インターコム，ファイル転送ユーティリティ「LAPLINK」の日本語化を完了
　アップル社・IBM社，1990年代に向けた技術構想を発表
　セガ・エンタープライゼスと日本ビクター，業務提携で基本合意
Programmer's Program「Final Super Pack」　134
オンラインソフト採集帳９「EXEXE.EXE」　136
HEALTHY? UNHEALTHY?⑨なぜ砂糖は歯に悪い？
ドラゴン・ポスト　138
付録ディスクのご案内　140
バーチャルスペースガイド　141
読者プレゼント

クイズ・ランダムスクリーン 142
オンラインサービス（SE-NET）のお知らせ
バックナンバーのご案内 143

付録 5"2HDフロッピーディスク
綴じ込み付録
PC-9801シリーズ・グラフィックスBIOS/LIO機能一覧

■1991年10月号

特集 286/386メモリ活用プログラミング
MS-DOSからプロテクトメモリを利用するために

EMS，XMSを中心に…
MS-DOSメモリ管理方式の歩み 加野島英渡 2
仮想記憶モードをもった高機能CPUの概要
80286，80386のアーキテクチャ 秋山登 13
386によるプロテクトモードプログラミングの実際
プログラム実行環境「P386」の作成 秋山登 28
プロテクトメモリ管理規約の概要
MS－DOSのメモリ拡張とVCPI，DPMI 菱博善 45
80286，80386のプロテクトモードを使う
DOSエクステンダ活用プログラミング 森誠 52
プログラミングはどう変わるのか
フリーエリアが広がったMS-DOS 5.0
インタビュー●マイクロソフト（株） 69

IC千夜一夜 第4回
初期のメモリ素子 ―遅延線メモリとコアメモリ― 虎岩登 78
MS-DOSツール・ラボ 第7回
FAST基本ライブラリ "コア" FORTHコンパイラの作成③FAST
加野島英渡 82
実践C塾 基礎テクニック篇 第4回
辞書順ソート 大野元久 94
98活用アセンブラ技法／番外編 第2回
書式およびデバッグのコツ 星野操 104

パソコン博物館 第10回
クリーンコンピュータ／MZ-80C 井戸禎光 74
プログラマの道具学 第10回
ジャンクフード 祐安重夫 76
情報処理技術者試験 第2種合格！トレーニング
第7回 プログラム流れ図 望月照夫・沖山貴裕 113
Book Review「SGML入門」 中村正三郎 118
本のアンテナ 古田島義和
佐々木哲のニューヨークシステム通信 120
西川和久の希望的観測 122
コンピュータの法律「プログラムの改造・その4」
熊谷信太郎 124
アキバ短信「メッセサンオーII」
プログラマの仕事場 第7回
株式会社システムハウスミルキーウェイ 126

INFORMATION PAGE 129
NEWS & NEW PRODUCTS 130
　クレオ，構造化プログラミングが可能な「TrueBASIC」を発売
　ライフボート，本格的COBOLインタープリタを発売
　創夢，X Window X11R5関連製品の提供・サービスを開始
　ラインズコーポレーション，教育用「Mind WAVE」を発売
　FREIA，LASERSHOT用C言語インターフェースライブラリを発売
　三洋電機，MS-DOS 5.0を国内初製品化，バンドルマシンを発売
　富士通，486SX搭載の高解像度デスクトップマシンを発売
　日本サン・マイクロシステムズ，デスクトップSPARCstationを強化
　エプソン，最大7台のSCSI機器接続可能なSCSIボードなどを発売
　スピリット，DOS/V用マルチスクリーンエディタを発売
　アスキー，パソコン用UNIX学習ソフトを低価格で発売
　ボーランド・インターナショナル，アシュトンテイト社を吸収合併
Programmer's Program「Mortice Kern Systemsツールキット」 134
オンラインソフト採集帳10「DENEC」 136
HEALTHY? UNHEALTHY?⑩Cばかりがなぜもてる？
ドラゴン・ポスト 138
付録ディスクのご案内 140
バーチャルスペースガイド 141
読者プレゼント
クイズ・ランダムスクリーン 142
オンラインサービス（SE-NET）のお知らせ
バックナンバーのご案内 143

付録 5"2HDフロッピーディスク
綴じ込み付録
EMSファンクションリクエスト（INT 67H）一覧

■1991年11月号

特集 Windowsプログラムの作り方
VisualBASIC，TPW，BC＋＋など新処理系登場!!

SDK＋MS-Cの標準的開発環境を理解するために 2
Windows APIはこうして使う 永井知己
Windowsでも威力を発揮するオブジェクト指向言語 21
Borland C＋＋ 大野元久
クラスライブラリを使ったオブジェクト指向プログラミング 41
Turbo Pascal for Windows 有澤雄志
ビジュアルなプログラミング環境を実現する 51
VisualBASIC 海老原浩之
ウィンドウ環境を生み出したGUIの元祖 60
Smalltalk/V Windows 1.1
クラスライブラリ，リソース作成，SQL処理など 64
Windowsプログラム開発支援ツール
　Actor 3.1 Professional
　C＋＋/Views
　SQLWindows

プログラム「遊」工房 十乃巻
「SNAKE」ゲーム 谷口典生 68
IC千夜一夜 第5回
メモリのIC化 ―D-RAM LSIの発展― 虎岩登 78
MS-DOSツール・ラボ 第8回
リングバッファの利用・起動記憶プログラム TIMERING.EXE
加野島英渡 82
実践C塾 基礎テクニック篇 第5回
プリンタ出力 大野元久 92
98活用アセンブラ技法 タイマ編 第1回
タイマBIOSの補強 星野操 102
PRODUCTS REVIEW

日本語TrueBASIC V3.0J 113

パソコン博物館 第11回
新技術の開拓者／FM8 井戸禎光 74
プログラマの道具学 第11回
コーヒー 祐安重夫 76
情報処理技術者試験 第2種合格！トレーニング
第8回 COBOL 白井儀巳・佐野有一 117
Book Review「プログラマのうちあけ話」 中村正三郎 122
本のアンテナ 古田島義和
佐々木哲のニューヨークシステム通信 124
西川和久の希望的観測 126
コンピュータの法律「プログラムの改造・その5」
熊谷信太郎 128
アキバ短信「万世本店」
プログラマの仕事場 第8回 マイクロプローズジャパン株式会社 130

INFORMATION PAGE 133
NEWS & NEW PRODUCTS 134
　ボーランドジャパン，98用Turbo Pascal for Windowsを発売
　サン・ソフト社，分散環境用システムソフトウェアSOLARISを発表
　東芝，J-3100対応のネットワーク用OSを発売
　エプソン，ハイコストパフォーマンスの16ビットデスクトップを発売
　富士通・PFU，RISCを採用したUNIXコンピュータ7機種を発売
　日本サン・マイクロシステムズ，SPARCstation初のラップトップモデルを発売
　日本エヌ・シー・アール，68040搭載のUNIXマシンを発売
　三菱電機，68040搭載の低価格CISCマシンを発売
　沖電気工業，LED光源，PostScript搭載のページプリンタを発売
　マイクロソフト，Windows 3.0対応のExcel，Wordを発表
　日本IBM，XGAの推進を発表，OADGなど対象に，
　アップルコンピュータ，アドビとフォント技術の協力体制を強化
Programmer's Program「ZZ386」 138
オンラインソフト採集帳11「BATCP.COM」 140
HEALTHY? UNHEALTHY?⑪胃が痛くなる話

ドラゴン・ポスト　142
付録ディスクのご案内　144
バーチャルスペースガイド　145
読者プレゼント
クイズ・ランダムスクリーン　146
オンラインサービス（SE-NET）のお知らせ
バックナンバーのご案内　147

付録　5"2HDフロッピーディスク
綴じ込み付録
日本語Microsoft Windows 3.0環境ファイル機能一覧

■1991年12月号

特集　やっぱり僕らはBASIC！
用途を選ばぬそのプログラミングパワーを探る

最先端の仕様を満足する美しいBASIC
データ構造＋プログラム構造＝ニューベーシック　市川新　2
N88BASICからQuickBASICへ
構造化プログラミングを学ぶ　塚本吉彦　11
プロジェクト指向環境を備えた先進のBASIC
Microsoft BASIC Ver.7.1　海老原浩之　16
512倍簡単になるWindowsプログラミング
Microsoft VisualBasic　酒井法雄　25
企業に生きるBASIC　50万ステップにおよぶBASICシステム
加藤雅彦　38
簡易性，多機種対応など教育用に適した特色
BASIC/98　木戸能史　40
生みの親が作った由緒正しい処理系
TrueBASIC　54

プログラム「遊」工房　拾壱乃巻
「JOKE/TOKEI」プログラム　谷口典生　74
IC千夜一夜　第6回
もうひとつの8ビットCPU　－MC6800－　虎岩登　80
MS-DOSツール・ラボ　第9回
フリーメモリ残量調整プログラム　REMMEM.EXE/RELMEM.EXE
加野島英渡　84
実践C塾　基礎テクニック篇　第6回
ちょっと便利な関数群　大野元久　94
98活用アセンブラ技法　タイマ編　第2回
時間取得・表示プログラム／プリンタスプーラの作成　星野操　101

付録　5"2HDフロッピーディスク
綴じ込み付録
QuickBASIC Ver.4.5ステートメント・関数一覧

パソコン博物館　第12回
はじめての16ビット機／MULTI-16　井戸禎光　70
プログラマの道具学　第12回
電子メール　祐安重夫　72
情報処理技術者試験　第2種合格！トレーニング
第9回　FORTRAN　稀代智明　113
Book Review「UNIX4.3BSDの設計と実装」　中村正三郎　118
本のアンテナ　古田島義和
SE-NET拾い読み　120
西川和久の希望的観測　122
コンピュータの法律「他人にプログラムを作成させた場合の著作権の帰属　その1」　熊谷信太郎　124
アキバ短信「ソフマップMidiLAND」
プログラマの仕事場　第9回　株式会社スタジオ・ゲン　126

INFORMATION PAGE　129
NEWS & NEW PRODUCTS　130
　データショウ'91，160余社が参加して開催される
　日本IBM，次世代の統合プラットフォーム「OS/2 J2.0」を発売
　マイクロウェア・システムズ，バージョンコントロールシステム「PVCS」を発売
　伊澄システム製作所，多機種対応の低価格アセンブラを発売
　日本IBM，日本初のXGA搭載デスクトップパソコンほかを発売
　東芝，DOS/Vに対応したDynaBook Vシリーズなどを発売
　日本電気，カラー液晶搭載した98NOTE，マルチメディアパソコンなどを発売
　アップルコンピュータ，ノートブック型パソコンなど新機種を発表
　エプソン，カラー液晶ディスプレイ搭載のブック型パソコンなどを発売
　IBM・アップル，テクノロジー分野での提携に合意
Programmer's Program「Newton-Backup」　134
オンラインソフト採集帳12「LHQ」　136
HEALTHY? UNHEALTHY?⑫目の疲れには……
ドラゴン・ポスト　138
付録ディスクのご案内　140
バーチャルスペースガイド　141
読者プレゼント
クイズ・ランダムスクリーン　142
オンラインサービス（SE-NET）のお知らせ
バックナンバーのご案内　143

Programmer's Page
MONTHLY MAGAZINE FOR PROGRAMMERS

月刊プログラマーズ・ページ バックナンバーのご案内

■定価　各980円

「月刊プログラマーズ・ページ」のバックナンバーは，ご注文いただければ，全国どこの書店でもお求めになれます（書店に届くまで1か月ほどかかります）。
小社に直接ご注文なさるときは，何月号かを明記の上，現金書留にて雑誌価格＋送料を下記までお送りください。送料は1部81円です（ただし7月号は1部86円）。2冊以上の場合は，号数・冊数により送料が異なりますので，小社までお問い合わせください。

〒102　東京都千代田区平河町2-4-14
平河町KSビル4階
(株)翔泳社　マーケティング部PP係宛
TEL 03-3263-0447

＊ 1991年1月号・2月号・3月号・4月号のバックナンバーは品切れです。

Programmer's Page
MONTHLY MAGAZINE FOR PROGRAMMERS

月刊プログラマーズ・ページ2月号
(1月18日発売)

付録: 5" 2HDフロッピーディスク付き!!

特集
通信ソフトの作り方

RS-232Cを使用したシリアル通信の概要からPC-9801のハードウェアを解説し，通信機能をもつソフトウェアに必要となるルーチンを実際にライブラリとして構築する．さらに，この通信ライブラリを用いた通信記述言語を作成して，簡単なターミナルソフトを実現する．

月刊プログラマーズ・ページ　1992年1月号
1992年1月1日発行

発行所　株式会社翔泳社
東京都千代田区平河町2-4-14平河町KSビル4F・5F
TEL 03-3263-0447　〒102
発行人　長廻健太郎
編集長　清水　隆
副編集長　鈴木嘉平
編集スタッフ　杉野貴美子＋上田純美礼＋芳山充弘
表紙デザイン　SE（翔泳社デザイン研究所）
広告スタッフ　相澤正夫＋澤本良知＋種村嘉彦
協力　根本裕三郎＋(株)ドモン・マインズ＋小林弘人(図版デザイン)
写植　株式会社東巧社＋株式会社トーヨー・プリント社
印刷　凸版印刷株式会社
定価　980円（本体951円，税29円）

© 1992 SHOEISHA
本書の一部または全部を著作権法の定める範囲をこえ，無断で複写，転載，テープ化，データファイル化することを禁じます．
＊本書の内容に関するご質問は，ご面倒でも必ず文書にて編集部までお問い合わせください．
＊本誌に記載した会社名，ハードウェア・ソフトウェアの商品名は，各社の商標または登録商標です．

EDITOR'S PAGE

●Windowsやマルチメディアなど，パソコン上でグラフィックスを扱う環境が話題になっている．しかし，写真や絵を美しく，しかも実用的なスピードで扱うには，当然のことながら，現在のハードウェアでは不足だ．より解像度を高くし，多くの色を表示するにはいくつもの関門がある．メモリ，CPU，DSPなどが速く，そして安くなること，画像圧縮や通信などに関わる規格が定められること，記憶媒体からの転送速度は？ディスプレイ装置は？……など，きりがない．どれもがいずれクリアされると思うのだが，16色の写真を出すのにこんなに時間がかかるなんて．(TA)

●近頃OS WARとやらが華やかに行われているらしい．DR DOS 6.0だとかNTだとかOS/2 2.0とかソラリスとか，はてはPinkとやらまで，じつに賑やかである．まぁ，DOS 5.0の次がなんになろうとかまわないけど，安くてバグが少なくて今あるツールがそのまま活用できるのがいいなぁ……と思うのは私だけかな．(嘉平)

●おかげさまで1周年．無事，発売日に遅れることなく（←当たり前だ！）1周年を迎えることができたのも，読者，著者の皆様はもちろんのこと，超人的なスケジュールをこなしていただいた写植，印刷会社，無理をきいていただいたデザイナー，イラストレーターの方々のおかげだと，心から感謝しています．2年目もよろしくお願いします．(ま)

●「初心者にもわかるようなプログラミング記事を！」という声に応えて，今月号から新連載「イカすBASICプログラミング入門」がはじまりました．「構造化プログラミングをカッコよくスマートに実現する」ためのプログラミング入門講座です．これからプログラムを学ぼうという方，すでにプログラミングを教える側の立場にたっているという方，第1回についてのご意見，ご感想などをぜひ編集部までお寄せください．(kos)

●今年もスキーのシーズンがやってきた．「でも何か欠けてる」「そうだ今年は車がない」「4駆の納車は早くて来年春だよ」「それ以前にお金がない」というわけですでに今シーズンは終っていた．あとは誰かが玄関まで迎えにきてくれることを期待するしかない．(Yoshi)

払込票

通常払込料金加入者負担

口座番号	東京 4-154493
加入者名	株式会社 翔泳社
金額	11760
払込人住所氏名	※
備考	
受付局日附印	

切り取らないで郵便局にお出しください。

記載事項を訂正した場合は、その箇所に訂正印を押してください。

払込通知票

料金負担

口座番号	4-154493
加入者名	株式会社 翔泳社
金額	11760
料金	払込み／特殊 円
備考	
受付局日附印	

番号

払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

正次氏が「ソフトウェア技術
のソフトウェアハウスを目指
成立した会社である。ご自身
の経験をもつだけに、技術者
てみるような経営姿勢を否定
その可能性を追求したいと考
環境を提供し、いい仕事をし
求しつづけてきた。
役の岡田謙二氏、PC事業部
橋誠氏を訪ね、パソコン部門
うかがった。

て、最近新たに加わったのが、
ンのパッケージソフト開発を中
うPC事業部。最初は特定業種向
ッケージからスタートして、今
分たちのオリジナルな商品を積
打ち出していこう、というのが
向性である。

しみな
ッケージソフトの開発

氏は、このPC事業部の今後を楽

月刊プログラマーズ・ページ 広告資料請求カード

8	会社(学校)名		9	所属(学部)		10	従業員数	

※下記の該当する番号をご記入ください。

11	業種		12	職種		13	パソコン利用状況	

業種 ①製造業 ②金融・保険・不動産業 ③商社・卸・小売業 ④建設業 ⑤通信・運輸 ⑥システム・ソフト ⑦その他サービス業 ⑧公務員 ⑨その他

職種 ①コンピュータ(ハード)技術者 ②コンピュータ(ソフト)技術者 ③エレクトロニクス技術者 ④プログラマ ⑤その他の技術者 ⑥専門職 ⑦事務・営業職 ⑧経営管理職 ⑨商工サービス ⑩教職 ⑪学生 ⑫その他

パソコン利用状況 ①会社で利用 ②個人で利用 ③会社でも個人でも利用 ④利用していない

●希望される広告資料　　年　　月号

希望項目 ＼ 広告資料請求No.												
A 購入希望	A	A	A	A	A	A	A	A	A	A	A	A
B 説明希望	B	B	B	B	B	B	B	B	B	B	B	B
C 資料希望	C	C	C	C	C	C	C	C	C	C	C	C

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

月刊プログラマーズ・ページ2月号

（1月18日発売）

付録：5″ 2HDフロッピーディスク付き!!

特集

通信ソフトの作り方

RS-232Cを使用したシリアル通信の概要からPC-9801のハードウェアを解説し，通信機能をもつソフトウェアに必要となるルーチンを実際にライブラリとして構築する．さらに，この通信ライブラリを用いた通信記述言語を作成して，簡単なターミナルソフトを実現する．

EDITOR'S PAGE

●Windowsやマルチメディアなど，パソコン上でグラフィックスを扱う環境が話題になっている．しかし，写真や絵を美しく，しかも実用的なスピードで扱うには，当然のことながら，現在のハードウェアでは不足だ．より解像度を高くし，多くの色を表示するにはいくつもの関門がある．メモリ，CPU，DSPなどが速く，そして安くなること，画像圧縮や通信などに関わる規格が定められること，記憶媒体からの転送速度は？ディスプレイ装置は？……など，きりがない．どれもがいずれクリアされると思うのだが，16色の写真を出すのにこんなに時間がかかるんて．（TA）

●近頃OS WARとやらが華やかに行われているらしい．DR DOS 6.0だとかNTだとかOS/2 2.0とかソラリスとか，はてはPinkとやらまで，じつに賑やかである．まぁ，DOS 5.0の次がなんになろうとかまわないけど，安くてバグが少なくて今あるツールがそのまま活用できるのがいいなぁ……と思うのは私だけかな．（嘉平）

●おかげさまで1周年．無事，発売日に遅れることなく（←当たり前だ！）1周年を迎えることができたのも，読者，著者の皆様はもちろんのこと，超人的なスケジュールをこなしていただいた写植，印刷会社，無理をきいていただいたデザイナー，イラストレーターの方々のおかげだと，心から感謝しています．2年目もよろしくお願いします．（ま）

●「初心者にもわかるようなプログラミング記事を！」という声に応えて，今月号から新連載「イチすBASICプログラミング入門」がはじまりました．「構造化プログラミングをカッコよくスマートに実現する」ためのプログラミング入門講座です．これからプログラムを学ぼうという方，すでにプログラミングを教える側の立場にたっているという方，

料金受取人払

麹町局承認

3910

差出有効期間
平成4年11月30日まで

切手不要

郵便はがき

1 0 2 - 0 0

東京都千代田区平河町2-4-14
平河町KSビル
株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係行

＊当社で記入

1 SE	2 受付No.

月刊プログラマーズ・ページ広告資料請求カード

3 会員番号

＊すでに会員番号をお持ちの方はご記入ください。

4 フリガナ 姓　名
氏名

5 生年月日　19　年　月　日

6 自宅住所 〒
電話（　　）

7 会社住所 〒
電話（　　）

資料送付希望先の番号に○印をつけてください。

払込票

通常払込料金加入者負担

口座番号 東京 4 ― 15493

加入者名 株式会社 翔泳社

金額 11760

払込人住所氏名 ※

備考

受付局日附印

切り取らないで郵便局にお出しください。

記載事項を訂正した場合は、その箇所に訂正印を押してください。

払込通知票

通常払込料金加入者負担

口座番号 東京 4 ― 15493

加入者名 株式会社 翔泳社

金額 11760

払込み 特殊 円

料金

備考

受付局日附印

払込人住所氏名 ※(郵便番号) TEL( )

各票の※印欄は、払込人において記載してください。

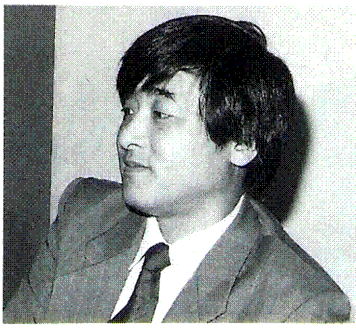
この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。(郵政省)

正次氏が「ソフトウェア技術
のソフトウェアハウスを目指
成立した会社である。ご自身
の経験をもつだけに、技術者
てみるような経営姿勢を否定
その可能性を追求したいと考
環境を提供し、いい仕事をし
求しつづけてきた。
役の岡田謙二氏、PC事業部
橋誠氏を訪ね、パソコン部門
うかがった。

ウェアハウスを下請け的発想でしか見ないのでは、ソフトウェア技術者の社会的な地位向上はありえない、というのが当社の基本にある考え方です。では下請け的発想や立場からどのように脱却するか、やはりユーザーに対して、トータルなコンサルテーションができるような力をつけることだと思います。それによって社会に貢献し、私たちの社会的地位を向上させていこうという姿勢が大切なんじゃないでしょうか。」

事実、同社は質の高い仕事を提供することで評価を高め、大和証券、横河

合的なネットワークのシステムの開発が進められている。日産自動車や鹿島建設など大手優良企業の総合的なシステムをクリエイトする仕事である。



岡田謙二氏

て、最近新たに加わったのが、
ンのパッケージソフト開発を中
うPC事業部。最初は特定業種向
ッケージからスタートして、今
分たちのオリジナルな商品を積
打ち出していこう、というのが
向性である。

しみな
ッケージソフトの開発

氏は、このPC事業部の今後を楽
見守っている一人だ。
業種向けのものとしては、レン
カーの大手8社が行っている
ネット″という大きなネットワ
ありまして、このネットの発注
システムや顧客管理システムなどを当社で手掛けています。

ただ、これはOEMのような形のものですから、吉正の名前はどこにも入っていません。そこで、当社独自のパッケージを作っていこうということで、最近、ライブラリ関係のパッケージの開発・販売を始めました。まだ規模は小さいですが、夢のある分野として期待しています。」

パッケージソフトとして発売されているのは、C言語でRS-232Cの各種制御を行うための『RS-232Cライブラ

定期購読のご案内

- ■定期購読は、1年間12冊と限定させていただきます。
- ■購読料のお支払は、この振替用紙をご利用ください。銀行振込、現金書留によるご送金は、ご遠慮ください。
- ■振替用紙に、ご住所、お名前をご記入のうえ、郵便局の貯金窓口にてお支払ください。
- ■小社より領収書の発行はいたしませんので、郵便局にて「郵便振替払込金受領証」をお受取ください。
- ■お申し込みいただいた方には、弊社より直接発送いたしますので、到着が発売日より遅れることがございますが、ご了承ください。

この欄は、加入者あての通信にお使いください。

月刊プログラマーズ・ページ　定期購読直接申し込み書

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

年　　月号より

年間購読料 11,760 円(税・送料込み)

■ご希望の購読開始月号をご記入ください。

■送付先（住所・氏名は正確にご記入ください）

住所	□□□-□□	フリガナ	
		氏名	
		TEL	

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。　（郵　政　省）

郵便はがき

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係行

2	受付No.	

マーズ・ページ広告資料請求カード

*すでに会員番号をお持ちの方はご記入ください。

名	5	生年月日	19　年　月　日

6	自宅住所	〒　電話　(　　)
7	会社住所	〒　電話　(　　)

資料送付希望先の番号に○印をつけてください。

Progr Pag

月刊プログラ
(1月18日発売)
付録: 5″ 2H
特集
通信ソ

RS-232Cを使用し
ハードウェアを解説
要となるルーチンを
らに，この通信ライ
て，簡単なターミナ

ちょっと気になるこの会社

吉正電子株式会社

吉正電子は、現社長の櫻井正次氏が「ソフトウェア技術者の社会的地位向上」と「真のソフトウェアハウスを目指す」ことを掲げ、昭和51年に成立した会社である。ご自身がソフトウェア技術者としての経験をもつだけに、技術者を単なる〝仕事の道具〟としてみるような経営姿勢を否定し、コンピュータが好きで、その可能性を追求したいと考える人たちに、いかに最適な環境を提供し、いい仕事をしてもらうか、ということを追求しつづけてきた。

東京・代々木の本社に取締役の岡田謙二氏、PC事業部の相沢伸一氏、浜光彦氏、髙橋誠氏を訪ね、パソコン部門での事業開発を中心にお話をうかがった。

ソフトウェア技術者の地位向上をめざす

求人誌などを見ると〝明日のコンピュータ社会を担う〟といったソフトウェアハウスのキャッチフレーズにはよくお目にかかるが、〝技術者のためのソフトウェアハウスを目指す〟というのはちょっとユニーク。2次外注からスタートして、すぐに1次外注、そして現在、自社開発のパッケージソフトを発売するまでになった吉正電子だが、「真のソフトウェアハウス」とは具体的に何を指すのか、まずはその点について、取締役の岡田氏にうかがった。

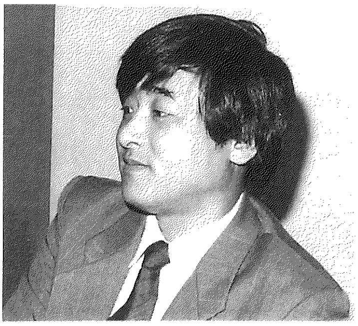
「ユーザーも、私たち自身も、ソフトウェアハウスを下請け的発想でしか見ないのでは、ソフトウェア技術者の社会的な地位向上はありえない、というのが当社の基本にある考え方です。では下請け的発想や立場からどのように脱却するか、やはりユーザーに対して、トータルなコンサルテーションができるような力をつけることだと思います。それによって社会に貢献し、私たちの社会的地位を向上させていこうという姿勢が大切なんじゃないでしょうか。」

事実、同社は質の高い仕事を提供することで評価を高め、大和証券、横河電機、日産自動車、鹿島建設など一流のユーザーと結びつき、1業績1社に絞り込んで徹底したシステム・コンサルテーションの実績を積み上げてきた。

そして現在、同社のソフトウェア事業は3つの柱で構成されている。

まず、汎用コンピュータ部門では、大和証券を中心とした証券システムの開発が中心だ。ここにはユーザーに出向いて開発を行っているチームと、ユーザーからラインを引いて社内で開発を行っているチームがあり、プランから開発まで一貫して行われている。

次に、社長自身が率いているプロジェクトがある。ここではUNIXマシンやVAXなどのミニコンを使った、総合的なネットワークのシステムの開発が進められている。日産自動車や鹿島建設など大手優良企業の総合的なシステムをクリエイトする仕事である。


岡田謙二氏

そして、最近新たに加わったのが、パソコンのパッケージソフト開発を中心に行うPC事業部。最初は特定業種向けのパッケージからスタートして、今後は自分たちのオリジナルな商品を積極的に打ち出していこう、というのがその方向性である。

楽しみなパッケージソフトの開発

岡田氏は、このPC事業部の今後を楽しみに見守っている一人だ。

「特定業種向けのものとしては、レンズメーカーの大手8社が行っている〝メガネット〟という大きなネットワークがありまして、このネットの発注システムや顧客管理システムなどを当社で手掛けています。

ただ、これはOEMのような形のものですから、吉正の名前はどこにも入っていません。そこで、当社独自のパッケージを作っていこうということで、最近、ライブラリ関係のパッケージの開発・販売を始めました。まだ規模は小さいですが、夢のある分野として期待しています。」

パッケージソフトとして発売されているのは、C言語でRS-232Cの各種制御を行うための『RS-232Cライブラ

リ』、C言語でISAMファイル処理を実現する『ISAMライブラリ』、C言語でJCA手順を実現する『JCA手順ライブラリ』、そしてオールグラフィックモードによる高速グラフィックウィンドウを実現する「グラフィックウィンドウライブラリ』の4種類。

「当社では、パッケージにソースコードを付けて販売しています。こうしたライブラリでもソース別売という形が多いと思いますが、私たちは敢えて、低価格でソースも付けて、それがたくさんの人にうまく利用されれば、という気持ちでやっています。

ライブラリの場合には、ユーザーのほとんどがソフトハウスやメーカーの方たちですから、ソフトハウスの方たちがこれを使ってより生産性を向上させたり、あるいはこれを改造して新しいパッケージ商品を作ったり、そういうことに役立つ商品づくりをめざしてみようと思っています。プロの人たちに役立つものを提供するには、WIndowsでもOS/2でも、自分たちが常にできるだけ先端の技術を持っていないといけないわけで、たんへんではありますが。PC事業部の若い人たちは意気込んでやっていますから、きっといいものを出していってくれるでしょう。」

92年1月にはOS/2対応の『ユーザーサポートシステム』を発売する予定だというが、同社のパッケージづくりの思想は、それ自体でも十分活用する価値を持ちながら、さらにユーザー自身が自由に拡張したり、変更して使うことのできる、これまでのパッケージとはひと味違ったスタイルのパッケージづくりにあるようだ。

0（ゼロ）から始める積極性が大切

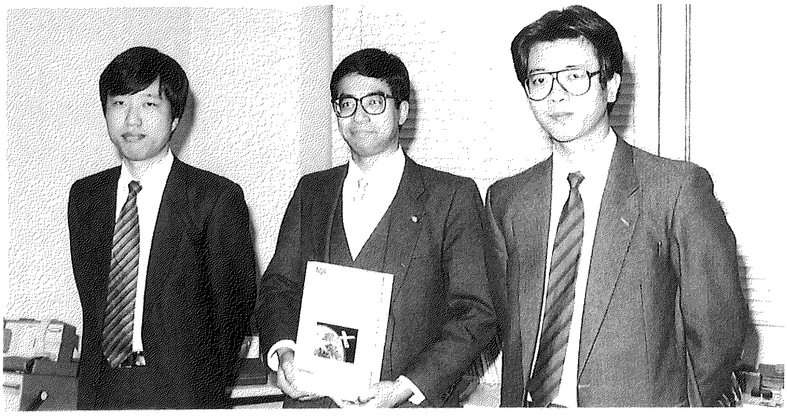
さて、同社では、新人の採用や育成についてはどんな考えを持っているのだろうか。

「当社の基本は先ほど述べたように、会社の体質を強くして、いちばんいい環境を提供して、余裕のある仕事をしてほしいということです。

そこで働く技術者に望むのは、積極性、好奇心など、自分から挑戦してみる、飛び込んでみるという精神でしょうね。

オリジナリティを持った仕事をしてきた会社ですから、それぞれの仕事も0（ゼロ）から始めるというケースが結構多いんです。たとえば今回のパッケージソフトの販売についても、広報宣伝の方法から販路、そしてパッケージのデザインなどにいたるまで、最初から社員が手探りで作り上げました。それはソフトウェア開発の仕事にも同様に当てはまることです。

0から試行錯誤を繰り返しながら、ものを作っていくことが好きな人、そのプロセスを楽しめる人に、一緒に仕事に加わってほしいと思います。」と岡田氏。



写真左から相沢伸一氏、髙橋誠氏、浜光彦氏

パッケージソフトの未来を夢見て

最後に、新しい部門として岡田氏が期待していると語っていたPC事業部の部屋を覗いてみることにした。ここで中心になって開発を担当しているのが、相沢伸一氏、浜光彦氏、髙橋誠氏の3人である。3人ともパッケージソフトは初めての経験で、受託システム開発などの仕事と比較すると、まったく違った種類の苦労と喜びがあると語ってくれた。

「パッケージソフトの場合には、必要な機能、誰にもわかりやすい操作性、使いやすいマニュアル類の完備など、商品としての完成度を求められますから、その点に気を使いますね。これから、まだまだクリアしなければならないことが多いと思っています。」

（相沢氏）

「パッケージソフトは、これが商品としてヒットしたら、という夢がありますね。その点は他の仕事と違った魅力だと思います。また、一般のユーザー向けということでプログラムの組み方ひとつにしてもしっかり作らなきゃいけないという緊張感がありました。自分にとってもたいへん勉強になりました。今後も試行錯誤を繰り返しながら、いろいろなものに挑戦していきたい。」

（浜氏）

「私はパソコン自体は個人的に好きでやってきたんですが、このPC事業部に入ってからは何ヶ月も触っていないんです。先輩たちのおかげで（笑）、職場の雰囲気もすごくリラックスできて、仕事を楽しんでやれる環境ですね。」

（高橋氏）

同社は社員100名程度と企業としては小規模だが、社長以下ソフトウェア技術者としての誇りと自由な精神を持っている会社である。そんな会社からどんなパッケージが飛び出してくるか、今後を楽しみにしたいと思う。

高水準の技術開発にはパーフェクト精神が根幹です。

ワンランク上の提案型ソフトウェアハウス＝エルテックスは、ユーザーや上司の求めている以上の開発を追求していく、我がままプロ集団の組織。常に高レベルの開発を行っていくためには、たえず外部の開発にアンテナを張りめぐらしておかなくてはなりません。その一環として、毎年数名、ラスベガスのコムデックスショー等、1週間程度の視察旅行に参加します。パーフェクトなものを求め続けていく技術者たちに、私たちは、たえず評価を与えていきます。

技術者随時募集

会社概要

ELTEX

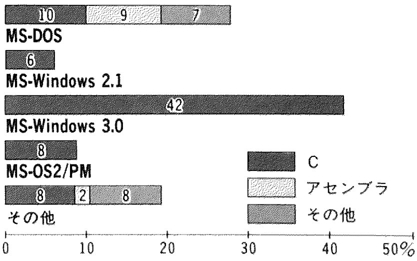
設　立	昭和60年12月14日
資本金	1,000万円
事業内容	パソコンを中心としたシステムの提案・開発・販売　①CD-ROM・LD など新しいメディアに対応したシステムの開発 ②ネットワークの端末システムの開発 ③特定業種向けCADの開発・販売 ④基本ソフトの開発
特　徴	大手クライアント向けのシステム開発が多く、ネットワーク関連システムやマルチメディアなど幅広く対応している。また、CADを中心とした自社プロダクツにも積極的で、その比率も増加している。基本ソフトの分野では、ハードウェア専門のベンチャー企業とジョイントし、OEM製品のBIOS・ユーティリティなどを企画から手掛けている。
取引先	三菱電機、三菱自動車、日本航空　他
関連会社	株式会社ソーテック、学校法人福井情報ビジネス総合学院

募集要項

ELTEX

職　種	①システムエンジニア　②プログラマー
資　格	専門卒以上　20～36歳の方　※C言語・アセンブラ言語経験者大歓迎　※未経験者でも好奇心・向学心の旺盛な方は歓迎します。
給　与	17万円（専門卒初任給）～46万円（36歳課長、諸手当含む） 24歳例：23万5000円（情報処理1種手当・住宅手当含む） 30歳例：31万6000円（中級SE、情報処理1種・住宅手当含む）
手　当	時間外手当、役職手当（5000円～6万円)、住宅手当（1万5000円)、資格手当（2種：5000～特種：2万円)、通勤手当
賞　与	賞与年2回（平成1年度実績年間5ヵ月)、決算賞与年1回（平成1年度実績1ヵ月）
勤務地	本社（横浜駅近く） ◇派遣・出向はありません。
勤務時間	フレックスタイム制（コアタイム10:00～16:00）試用期間中は標準時間9:00～17:45
休日休暇	完全週休2日制（土・日)、祝日、年3回連休（GW・夏期・年末年始)、有給・特別・休暇
福利厚生	各種社会保険完備、退職金制度、財形貯蓄、社員旅行、海外研修、定期健康診断、社員寮、スポーツクラブ法人会員
応募方法	電話連絡の上、履歴書(写真貼付)、職務経歴書を持参し来社ください。郵送も受け付けます。
連絡先	〒220 横浜市西区南幸二丁目20番5号　東伸24ビル5階　℡(045)322-9667　総務課

最近のOS・言語別開発動向

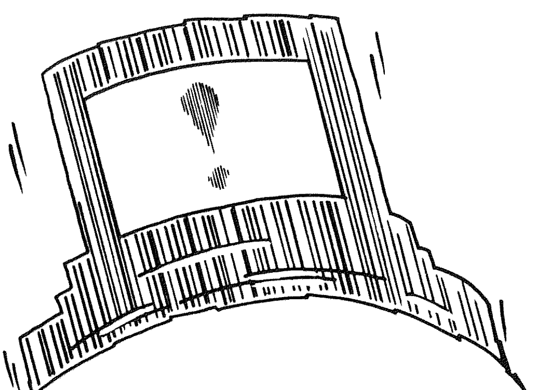


採用HOT・LINE
TEL.(045)322-9667

株式会社エルテックス

美光亭志寿天夢曰く、あっ！と驚かせる快感を味わいたい。

フレキシブルなフィールド、高いレベルの技術力。ふたつの条件が満たされているから、ユニークな発想が活かされるのだ。着想と技術の二人三脚で、これもやりたい、あれも作ってみたいという夢が、実現できる。その発表は、コンピュータ社会という舞台。観客は常に新しいものを期待しているユーザーだ。今、出囃子が鳴り、舞台へのぼる時がきた。



●美光システムワークス。3つのユニークな特徴

その1：創立者は、フリーで活躍していたエンジニア
フリーでシステム設計などを手がけるには、しっかりしたノウハウやユーザーからの厚い信頼がなくてはなりません。それがハードウェア資源を最大限に活かした高性能システム開発の基盤となっています。

その2：専門の営業部を設けてある
ニーズとダイレクトで接することができ、そこから活気が生まれ、開発意欲にフィードバックされて、より活性化をはかります。

その3：新会社ならではの、イノベーショナルな体勢
〝変化を友として、自らも変身し、共に栄えよう〟のモットーに、ソフトウェア開発に着手。自由なフィールドと技術力の高さの相乗効果で、次世代コンピュータ開発にも、積極的に参画しています。
※広い情報化社会に、あなた独自の着想で、波紋を投げかけてください。

■事業内容■

- ●マイクロプロセッサ応用機器用組み込みソフトウェアの開発
- ●ROM化されるソフトウェアの設計・製造、検査保守
- ●パソコン応用システム用ソフトウェア
- ●ハード、ソフトを含めたシステム設計開発
- ●拡張保守

■職種■

- ●SE・プログラマー
- ●マイコン応用機器ソフト開発技術者
- ●システム営業

※未経験者も可

学生の方の資料請求歓迎

■会社概要■

設立／昭和62年4月21日　**資本金**／1000万円
代表者／山本　侑　**社員数**／8名
年商／1億6000万円（2年度実績）
事業内容／マイクロプロセッサ応用機器用組み込みソフトウェアの開発、ROM化されるソフトウェアの設計・製造・検査保守、パーソナルコンピュータ応用システム用ソフトウェア、ハードウェア・ソフトウェアを含めたシステム設計開発、拡張保守
開発実績／●FDデュプリケータ　●FDエバリュエータ　●プリント基板配線編集システム　●FDプロテクションソフト　●トラックポジション測定装置

■募集要項■

職種　●SE・プログラマー
●マイコン応用機器ソフト開発技術者
●システム営業
資格　高卒以上、30歳位迄　経験者／35歳迄
給与　経験・年齢を考慮の上、優遇します。
固定給＋歩合給制　大卒初任給／19万円
待遇　昇給年1回、賞与年2回（62年実績5ヵ月）、交通費支給、各種社会保険完備、退職金・財形貯蓄制度有、諸手当（営業・職能・家族・役職）有、社内旅行（年2回）、クラブ活動（ゴルフ、野球etc.）
勤務地　東京
勤務時間　9：15～17：30
※フレックスタイム制
休日休暇　日曜、祝日、第1・3土曜、夏季、年末年始、有給、他
応募　電話連絡の上、履歴書（写真貼付）を持参ください。
交通　地下鉄日比谷線・都営浅草線人形町駅下車徒歩3分

有限会社 美光システムワークス
〒103 東京都中央区日本橋富沢町4-6 糸重富沢町ビル4階
美光システムワークス　Tel03-3663-7822㈹　担当／山本

ANSI C関数一覧

●abort()　プログラムを直ちに終了する
```
#include <stdlib.h>
void abort(void);
```
●abs()　整数の絶対値を計算する
```
#include <stdlib.h>
int abs(int n);
```
●acos()　逆コサインを計算する
```
#include <math.h>
double acos(double arg);
```
●asctime()　日時文字列に変換する
```
#include <time.h>
char *asctime(const struct tm *timestruct);
```
●asin()　逆サインを計算する
```
#include <math.h>
double asin(double arg);
```
●atan()　逆タンジェントを計算する
```
#include <math.h>
double atan(double arg);
```
●atan2()　2つの値を除算し,逆タンジェントを計算する
```
#include <math.h>
double atan2(double num, double den);
```
●atexit()　プログラム終了の際に実行する関数を登録する
```
#include <stdlib.h>
int atexit(void (*function)(void));
```
●atof()　文字列を浮動小数に変換する
```
#include <stdlib.h>
double atof(const char *string);
```
●atoi()　文字列を整数に変換する
```
#include <stdlib.h>
int atoi(const char *string);
```
●atol()　文字列をlong型の整数に変換する
```
#include <stdlib.h>
long int atol(const char *string);
```
●bsearch()　配列内をバイナリサーチする
```
#include <stdlib.h>
void *bsearch(const void *item, const void *array, size_t number, size_t size,
int (*comparison)(const void *, const void *));
```
●calloc()　動的にメモリを割り当て,0に初期化する
```
#include <stdlib.h>
void *calloc(size_t, count, size_t size);
```
●ceil()　値より大きな整数のうち最小値を得る
```
#include <math.h>
double ceil(double z);
```
●clearerr()　ストリームのエラー状態指示子をクリアする
```
#include <stdio.h>
void clearerr(FILE *fp);
```
●clock()　プログラム起動から現在までの時間を得る
```
#include <time.h>
clock_t clock(void);
```
●cos()　コサインを計算する
```
#include <math.h>
double cos(double radian);
```
●cosh()　双曲線コサインを計算する
```
#include <math.h>
double cosh(double value);
```
●ctime()　暦時間を文字列に変換する
```
#include <time.h>
char *ctime(const time_t *timeptr);
```
●difftime()　2つの時刻の差を計算する
```
#include <time.h>
double difftime(time_t newtime, time_t oldtime);
```
●div()　整数を除算する
```
#include <stdlib.h>
div_t div(int numerator, int denominator);
```
●exit()　後始末をしてプログラムを終了する
```
#include <stdlib.h>
void exit(int status);
```
●exp()　指数関数を計算する
```
#include <math.h>
double exp(double z);
```
●fabs()　浮動小数の絶対値を計算する
```
#include <math.h>
double fabs(double z);
```
●fclose()　ストリームをクローズする
```
#include <stdio.h>
int fclose(FILE *fp);
```
●feof()　ストリームのファイル終端状態指示子を調べる
```
#include <stdio.h>
int feof(FILE *fp);
```
●ferror()　ストリームのエラー状態を調べる
```
#include <stdio.h>
int ferror(FILE *fp);
```
●fflush()　出力バッファをフラッシュする
```
#include <stdio.h>
int fflush(FILE *fp);
```
●fgetc()　ストリームから1文字読み込む
```
#include <stdio.h>
int fgetc(FILE *fp);
```
●fgetpos()　ファイル位置指示子の値を得る
```
#include <stdio.h>
int fgetpos(FILE *fp, fpos_t *position);
```
●fgets()　ストリームから1行読み込む
```
#include <stdio.h>
char *fgets(char *string, int n, FILE *fp);
```
●floor()　値を越えない最大の整数を得る
```
#include <math.h>
double floor(double z);
```
●fmod()　剰余を計算する
```
#include <math.h>
double fmod(double number, double divisor);
```
●fopen()　入出力用にストリームをオープンする
```
#include <stdio.h>
FILE *fopen(const char *file, const char *mode);
```
●fprintf()　フォーマットを指定したテキストをストリームに出力する
```
#include <stdio.h>
int fprintf(FILE *fp, const char *format, ...);
```
●fputc()　文字をストリームに書き込む
```
#include <stdio.h>
int fputc(int character, FILE *fp);
```
●fputs()　文字列をストリームに出力する
```
#include <stdio.h>
int fputs(const char *string, FILE *fp);
```
●fread()　ストリームからデータを読み込む
```
#include <stdio.h>
size_t fread(void *buffer, size_t size, size_t n, FILE *fp);
```
●free()　動的に割り当てたメモリを解放する
```
#include <stdlib.h>
void free(void *ptr);
```
●freopen()　ストリームをオープンしなおす
```
#include <stdio.h>
FILE *freopen(const char *file, const char *mode, FILE *fp);
```
●frexp()　浮動小数を仮数部と指数部に分ける
```
#include <math.h>
double frexp(double value, int *exp);
```
●fscanf()　ストリームからデータを読み込み解釈する
```
#include <stdio.h>
int fscanf(FILE *fp, const char *format, ...);
```
●fseek()　ファイル位置指示子をセットする
```
#include <stdio.h>
int fseek(FILE *fp, long int offset, int whence);
```
●fsetpos()　ファイル位置指示子をセットする
```
#include <stdio.h>
int fsetpos(FILE *fp, const fpos_t *position);
```
●ftell()　ファイル位置指示子の値を得る
```
#include <stdio.h>
long int ftell(FILE *fp);
```
●fwrite()　データをストリームに書き込む
```
#include <stdio.h>
size_t fwrite(const void *buffer, size_t size, size_t n, FILE *fp);
```
●getc()　ストリームから1文字読み込む
```
#include <stdio.h>
int getc(FILE *fp);
```
●getchar()　標準入力から1文字読み込む
```
#include <stdio.h>
int getchar(void);
```
●getenv()　環境変数を読み込む
```
#include <stdlib.h>
char *getenv(const char *variable);
```
●gets()　標準入力から文字列を読み込む
```
#include <stdio.h>
char *gets(char *buffer);
```
●gmtime()　暦時間を世界標準時に変換する
```
#include <time.h>
struct tm *gmtime(const time_t *caltime);
```
●isalnum()　文字が数字またはアルファベットかどうかをチェックする
```
#include <ctype.h>
int isalnum(int c);
```
●isalpha()　文字がアルファベットかどうかをチェックする
```
#include <ctype.h>
int isalpha(int c);
```
●iscntrl()　文字が制御文字かどうかをチェックする
```
#include <ctype.h>
int iscntrl(int c);
```
●isdigit()　文字が数字かどうかをチェックする
```
#include <ctype.h>
int isdigit(int c);
```
●isgraph()　文字が印字可能かどうかをチェックする(空白文字は除く)
```
#include <ctype.h>
int isgraph(int c);
```
●islower()　文字が小文字かどうかをチェックする
```
#include <ctype.h>
int islower(int c);
```
●isprint()　文字が印字可能かどうかをチェックする
```
#include <ctype.h>
int isprint(int c);
```
●ispunct()　文字が区切り文字かどうかをチェックする
```
#include <ctype.h>
int ispunct(int c);
```
●isspace()　文字が空白文字かどうかをチェックする
```
#include <ctype.h>
int isspace(int c);
```
●isupper()　文字が大文字かどうかをチェックする
```
#include <ctype.h>
int isupper(int c);
```
●isxdigit()　文字が16進数かどうかをチェックする
```
#include <ctype.h>
int isxdigit(int c);
```
●labs()　long型整数の絶対値を計算する
```
#include <stdlib.h>
long int labs(long int n);
```
●ldexp()　仮数部と2の指数部から浮動小数を得る
```
#include <math.h>
double ldexp(double number, int exp);
```
●ldiv()　long型整数を除算する
```
#include <stdlib.h>
ldiv_t ldiv(long int numerator, long int denominator);
```
●localeconv()　構造体lconvの初期化
```
#include <locale.h>
struct lconv *localeconv(void);
```
●localtime()　暦時間を現地時間に変換する
```
#include <time.h>
struct tm *localtime(const time_t *timeptr);
```
●log()　自然対数を計算する
```
#include <math.h>
double log(double z);
```
●log10()　常用対数を計算する
```
#include <math.h>
double log10(double z);
```
●longjmp()　大域ジャンプする
```
#include <setjmp.h>
void longjmp(jmp_buf environment, int rval);
```
●malloc()　動的メモリ(dynamic memory)を割り当てる
```
#include <stdlib.h>
void *malloc(size_t size);
```
●mblen()　マルチバイト文字の文字列の長さを得る
```
#include <stdlib.h>
int mblen(const char *address, size_t number);
```

●mbstowcs()　マルチバイト文字の文字列をwide文字の文字列に変換する
```
#include <stdlib.h>
size_t mbstowcs(wchar_t *widechar, const char *multibyte, size_t number);
```
●mbtowc()　マルチバイト文字をwide文字に変換する
```
#include <stdlib.h>
int mbtowc(wchar_t *charptr, const char *address, size_t number);
```
●memchr()　文字をメモリ領域の中から探す
```
#include <string.h>
void *memchr(const void *region, int character, size_t n)
```
●memcmp()　2つの領域を比較する
```
#include <string.h>
int memcmp(const void *region1, const void *region2, size_t count);
```
●memcpy()　メモリ領域をほかにコピーする
```
#include <string.h>
void *memcpy(void *region1, const void *region2, size_t count);
```
●memmove()　領域をそれと可能性のある領域に移す
```
#include <string.h>
void *memmove(void *region1, const void *region2, size_t count);
```
●memset()　領域をある文字で満たす
```
#include <string.h>
void *memset(void *buffer, int character, size_t count);
```
●mktime()　要素別の時間(broken-down time)を暦時間に変換する
```
#include <time.h>
time_t mktime(struct tm *timeptr);
```
●modf()　実数を整数部と小数点以下の部分とに分ける
```
#include <math.h>
double modf(double value, double *iptr);
```
●perror()　エラーメッセージを標準エラー出力へ出力する
```
#include <stdio.h>
void perror(const char *string);
```
●pow()　zのx乗を計算する
```
#include <math.h>
double pow(double z, double x);
```
●printf()　フォーマットを指定してテキストを標準出力ストリームに出力する
```
#include <stdio.h>
int printf(const char *format, ...);
```
●putc()　ストリームに文字を書き込む
```
#include <stdio.h>
int putc(int character, FILE *fp);
```
●putchar()　標準出力ストリームに文字を書き込む
```
#include <stdio.h>
int putchar(int character);
```
●puts()　標準出力ストリームに文字列を出力する
```
#include <stdio.h>
int puts(const char *string);
```
●qsort()　配列をソートする
```
#include <stdlib.h>
void qsort(void *array, size_t number, size_t size,
int (*comparison)(const void *, const void *));
```
●raise()　シグナルを送る
```
#include <signal.h>
int raise(int signal);
```
●rand()　疑似乱数を発生する
```
#include <stdlib.h>
int rand(void);
```
●realloc()　動的メモリを再割り当てする
```
#include <stdlib.h>
void *realloc(void *ptr, size_t size);
```
●remove()　ファイルを削除する
```
#include <stdio.h>
int remove(const char *filename);
```
●rename()　ファイル名を変更する
```
#include <stdio.h>
int rename(const char *old, const char *new);
```
●rewind()　ファイル位置指定子をリセットする
```
#include <stdio.h>
void rewind(FILE *fp);
```
●scanf()　標準入力ストリームからテキストを読み込み解釈する
```
#include <stdio.h>
int scanf(const char *format, ...);
```
●setbuf()　自分用のストリームバッファを設定する
```
#include <stdio.h>
void setbuff(FILE *fp, char *buffer);
```
●setlocale()　プログラムのロケール情報を設定し,検索する
```
#include <locale.h>
char *setlocale(int portion, const char *locale);
```
●setvbuf()　自分用のストリームバッファを設定する
```
#include <stdio.h>
int setvbuf(FILE *fp, char *buffer, int mode, size_t size);
```
●signal()　シグナルが検出された場合の処理を設定する
```
#include <signal.h>
void (*signal(int signame, void (*function)(int)))(int);
```
●sin()　サインを計算する
```
#include <math.h>
double sin(double radian);
```
●sinh()　双曲線サインを計算する
```
#include <math.h>
double sinh(double value);
```
●sprintf()　フォーマットを指定して文字列にテキストを書き込む
```
#include <stdio.h>
int sprintf(char *string, const char *format, ...);
```
●sqrt()　数値の平方根を計算する
```
#include <math.h>
double sqrt(double z);
```
●srand()　疑似乱数発生ルーチンにシードを与える
```
#include <stdlib.h>
void srand(unsigned int seed);
```
●sscanf()　文字列を読み込んでテキストを解釈する
```
#include <stdio.h>
int sscanf(const char *sring, const char *format, ...);
```
●strcat()　文字列を連結する
```
#include <string.h>
char *strcat(char *string1, const char *string2);
```
●strchr()　文字列から文字を検索する
```
#include <string.h>
char *strchr(const char *string, int character);
```
●strcmp()　文字列を比較する
```
#include <string.h>
int strcmp(const char *string1, const char *string2);
```
●strcoll()　ロケール情報を使って文字列を比較する
```
#include <string.h>
int strcoll(const char *string1, const char *string2);
```
●strcpy()　文字列をコピーする
```
#include <string.h>
char *strcpy(char *string1, const char *string2);
```
●strcspn()　文字列を比較し,一方にない文字からなる部分の長さを得る
```
#include <string.h>
size_t strcspn(const char *string1, const char *string2);
```
●strerror()　エラー番号を文字列に変換する
```
#include <string.h>
char *strerror(int error);
```
●strftime()　ロケール固有の時刻を表示形式にしたがって変換する
```
#include <time.h>
size_t strftime(char *string, size_t maxinum, const char *format,
const struct tm *brokentime);
```
●strlen()　文字列の長さを得る
```
#include <string.h>
size_t strlen(const char *string);
```
●strncat()　文字列の一部を他方に付け加える
```
#include <string.h>
char *strncat(char *string1, const char *string2, size_t count);
```
●strncmp()　文字列の一部を比較する
```
#include <string.h>
int strncmp(const char *string1, const char *string2, size_t count);
```
●strncpy()　文字列の一部をほかの文字列にコピーする
```
#include <string.h>
char *strncpy(char *string1, const char *string2, size_t count);
```
●strpbrk()　一方の文字列に含まれる文字が他方の文字列の最初に出てくる位置を得る
```
#include <string.h>
char *strpbrk(const char *string1, const char *string2);
```
●strrchr()　文字列中で,ある文字があらわれる最後の位置を得る
```
#include <string.h>
char *strrchr(const char *string, int character);
```
●strspn()　文字列を比較し,一方の文字列に含まれる文字からなる部分の長さを得る
```
#include <string.h>
size_t strspn(const char *string1, const char *string2);
```
●strstr()　文字列中から任意の文字列を検索する
```
#include <string.h>
char *strstr(const char *string1, const char *string2);
```
●strtod()　文字列を浮動小数に変換する
```
#include <stdlib.h>
double strtod(const char *string, char **tailptr);
```
●strtok()　文字列をトークンに分割する
```
#include <string.h>
char *strtok(char *string1, const char *string2);
```
●strtol()　文字列をlong型の数値に変換する
```
#include <stdlib.h>
long int strtol(const char *sptr, char **tailptr, int base);
```
●strtoul()　文字列をunsigned long型の数値に変換する
```
#include <stdlib.h>
unsigned long int strtoul(const char *sptr, char **tailptr, int base);
```
●strrxfrm()　文字列を変換する
```
#include <string.h>
size_t strxfrm(char *string1, const char *string2, size_t count);
```
●system()　ほかのプログラムを実行する
```
#include <stdlib.h>
int system(const char *program);
```
●tan()　タンジェントを計算する
```
#include <math.h>
double tan(double radian);
```
●tanh()　双曲線タンジェントを計算する
```
#include <math.h>
double tanh(double value);
```
●time()　現在の暦時間を得る
```
#include <time.h>
time_t time(time_t *tp);
```
●tmpfile()　一時ファイルを作成する
```
#include <stdio.h>
FILE *tmpfile(void);
```
●tmpnam()　一時ファイルの名前を作成する
```
#include <stdio.h>
char *tmpnam(char *name);
```
●tolower()　小文字に変換する
```
#include <ctype.h>
int tolower(int c);
```
●toupper()　大文字に変換する
```
#include <ctype.h>
int toupper(int c);
```
●ungetc()　入力ストリームに1文字戻す
```
#include <stdio.h>
int ungetc(int character, FILE *fp);
```
●vfprintf()　フォーマットを指定してテキストをストリームに書き込む
```
#include <stdarg.h>
#include <stdio.h>
int vfprintf(FILE *fp, const char *format, va_list arguments);
```
●vprintf()　フォーマットを指定してテキストを標準出力ストリームに書き込む
```
#include <stdarg.h>
#include <stdio.h>
int vprintf(const char *format, va_list arguments);
```
●vsprintf()　フォーマットを指定してテキストを文字列に書き込む
```
#include <stdarg.h>
#include <stdio.h>
int vsprintf(char *string, const char *format, va_list arguments);
```
●wcstombs()　wide文字の列をマルチバイト文字列に変換する
```
#include <stdlib.h>
size_t wcstombs(char *multibyte, const wchar_t *widechar, size_t number);
```
●wctomb()　wide文字をマルチバイト文字に変換する
```
#include <stdlib.h>
int wctomb(char *string, wchar_t widecharacter);
```

LIFEBOAT

アプリケーション開発のために正確な内部情報を入手できれば問題ありません。しかし、複雑化／大規模化の道をたどるシステム・リソースは、今まさにブラックボックスとなりつつあります。SOURCER〈ソーサー〉は、最も貴重な内部情報を解析&公開する世界で最も優れた逆アセンブラ。様々な実行ファイルやメモリから詳細なコメント付アセンブラソースリストを生成します。

ブラックボックスを解放する
世界最高水準 逆アセンブラSOURCER

不可能を可能にするSOURCER

■プログラムの内部動作の調査
■バグを含むプログラムの修正
■ドキュメントされていない機能の発見
■ソースコードの入手
■ウィルスの検出

SOURCER

SOURCERに内蔵されたデータアナライザとコードシミュレータは、複雑に区切られたセグメントを解析。インタラプト、サブファンクション、I／Oポートなどに関する手続きにコメントを付加。読みやすく、そのまま再アセンブル可能なソースリストを生成します。また、環境設定ファイルでSOURCERの解析方法をカスタマイズ可能。ユーザ独自のラベル定義や効果的なデータ・タイプなどを設定できます。

SOURCER
Commenting Disassembler

標準価格：¥38,000

最高9パスによる高品質な解析を実現。8086から最新の486まで各種CPUのインストラクションに完全対応します

SOURCERのコード解析は2から最高9までのきめ細かい解析パスによって行われます。実際に命令を実行（シミュレート）することにより、従来の逆アセンブラでは得られなかった高度な解析を可能としています。内部シミュレータは複数のセグメントに分散されたコードおよびデータを様々な解析手法を用いて分解します。インストラクションは8088から80486、8087から80387、V20／V30までフルに対応します。また、80286から80486までのプロテクトモードインストラクションも解析可能です。

SOURCERで出力されたソースファイルは、現在最も多く使われている下記のアセンブラ処理系と互換性があります

●Microsoft MASM 4.0／5.0／5.1　●OPTASM
●Borland TASM 1.0／2.0

C、FORTRAN、PASCALなど、言語に依存することなく様々なアプリケーションを解析します

●COMファイル／デバイスドライバ
●EXEファイル(EMS使用の場合は最大約420Kの大規模アプリケーションを解析可能)
●オーバーレイファイル
●任意のROMやRAMにロケーションされたコード(BIOSなど)

オリジナルよりも読みやすいアセンブラリストを出力。SOURCERは各種コメントを自動的に埋め込み／生成します

●プログラムのヘッダー情報
●各種ファンクションコールの機能と各レジスタの意味
●input／outputポートの操作状況
●アクセスしているデータの配置と値に関する情報
●複雑なコードの処理内容
●EMS4.0の割り込み
●その他多くのコメントをサポート

さらに高度な解析が可能。SOURCERはユーザカスタマイズ機能により解析方法を細部にわたって設定できます。

定義ファイル(.DEF)により、SOURCERの解析手法を細かく設定することが可能。解析するファイルやメモリの定義、データロケーションやサブルーチンに対する参照の指定などにより、さらに高度な解析が行えます。

●開始セグメントとオフセットの定義(ROMなどに対して)
●セグメントタイプ(CODE, DATA, STACKなどの分類)
●内容に即したラベルのユーザ定義
●データ項目のタイプ(DB, DWなど)と長さの指定など

その他の特徴

●各種アセンブラに対応したニーモニック表記
●インデックスコール／ジャンプの自動解析
●デバイスドライバおよびチェインされたデバイスドライバの検出、およびエントリポイントの検出
●レジデントプログラムのエントリポイントの検出
●出現したアイテムがどこで定義されているのかをレポートするクロスレファレンス
●アセンブラの不具合に対応した自動修正など

■対応機種………PC-9801シリーズ、FMRシリーズ、J-3100シリーズ、Panacom Mシリーズ、PS55シリーズ、AX
■対応DOS………日本語MS-DOS Ver. 2／3／4、DOS／V J4.0
■必要メモリ………640Kバイト以上
■パッケージ内容…5インチまたは3.5インチフロッピーディスク1セット。完全和文マニュアル
＊その他SOURCERの国内正式ユーザ登録カード含む

【お求め方法】●SOURCERのご用命は、お近くのパソコンソフト販売店へ。

株式会社ライフボート
〒101 東京都千代田区神田錦町3-6
PHONE:03-3293-4711(代) FAX:03-3293-4710

●Copyright ＊SOURCERは(米)V Communications, Inc.の登録商標です ＊その他、プログラム名、CPU名、システム名は一般に各メーカの(登録)商標です。
＊市販のソフトウェアは、法律により著作権が保護されています 逆アセンブル／解析の禁止されたソフトウェアに対しては、SOURCERを使用しないで下さい。

〈資料請求番号P03〉

BORLAND
The Leader in Object-Oriented Programming

問：Windows™ アプリケーション開発に最も有効なコンパイラは、何ですか？

（解答はページ下をご覧下さい）

(Inactive C:¥WORK¥BORLAND.EXE)

	従来の開発環境	？
Cコンパイラ	¥ 98,000	¥ 98,000
C++コンパイラ	なし	含む
Windows AP作成ツール	¥ 98,000	含む
アセンブラ	¥ 40,000	含む
合計	¥236,000	¥ 98,000

※上記の価格には、消費税は含まれていません。

パソコン上に、視覚的で快適な操作環境を提供するWindows 3.0。そこで利用するアプリケーション作成のためのコンパイラは、現在のものでベストといえるでしょうか？機能、スピード、価格…。これらの面で理想的ともいえるコンパイラが、登場しました。「BORLAND C++」です。BORLAND C++は、Windowsアプリケーションを開発、デバックするC/C++コンパイラとツールを標準で装備しています。

■Windowsアプリケーション開発をサポート

BORLAND C++は、Windowsプログラムを完全にサポート。高速かつ高性能な処理を実現しています。MDI(Multiple Document Interface)やDLLs(Dynamic Link Libraries)といったようなWindowsの機能を使って、大規模なプロテクトモードアプリケーションの開発が可能になります。さらに、Microsoftのリソースコンパイラと WINDOWS.H、Resource Work Shop(日本語版)が標準装備されていますので、別途の開発キット(SDK)は不要です。

■ANSI C／AT&T C++両用コンパイラ

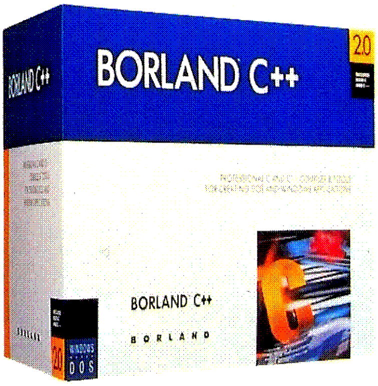
BORLAND C++は、CとC++両方のコンパイラとして利用できます。さらに、再コンパイルの速度を大幅に向上させる「コンパイル済みヘッダ」機能をサポート。MS-DOSの640KBというメモリの制限を破るオーバーレイマネージャVROOMの利用や、Turbo Driveによってプロテクトモードで動作する統合開発環境での大規模なプログラムの作成も可能です。

■Programmer's Platform

Programmer's Platformは、好きなツールを組み込む機能を持つ強力な統合開発環境で、MS-DOSやWindowsアプリケーション開発に必要なすべてのツールがすでに組み込まれています。さらにオーバーラップウインドウやマウス、そしてWindowsのプログラミングリファレンスを含んだオンラインヘルプをサポート。内蔵のマルチファイルエディタは、アンドゥ(実行の取消)／リドゥ(再実行)やマクロ機能を装えています。

■動作環境：PC-9801シリーズ(ハイレゾモード対応、LTは除く)、PC-286/386シリーズ／メモリ：640KB以上／MS-DOS：Ver 3.0以上／MS-Windows 3.0以上／マウス：バスマウス／要ハードディスク

■価格……………………………¥98,000(税別)



ユーザ向け情報誌
LANGUAGE EXPRESSプレゼント!!

■この広告をご覧になって、製品に関する資料を請求していただいた方全員に、ボーランド発行の情報誌「LANGUAGE EXPRESS」をさし上げます。

(答) BORLAND® C++

※All Borland Products are trademarks or registered trademarks of Borland International, Inc ©Borland International, Inc.
※Microsoft、Windowsは米国マイクロソフト社の登録商標です。その他、商品名は一般に各社の商標です。
●製品に関するお問い合せは、総販売元㈱マイクロソフトウェアアソシエイツ TEL 03-3486-1411までお願いします。

ボーランド株式会社
〒107 東京都港区南青山7-8-1 小田急南青山ビル TEL.03-3486-1400㈹
※社名変更いたしました。

〈資料請求番号P04〉

T1017777010988

雑誌コード17777-01